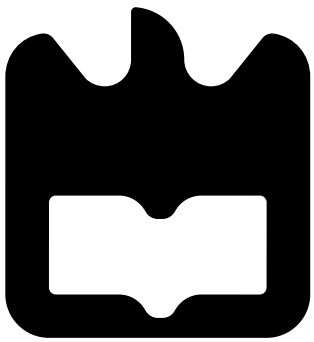
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2020

Carlos Manuel Lopes Soares

Desenvolvimento de um editor para criação de automações em assistentes de casas inteligentes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Professor Doutor Diogo Nuno Pereira Gomes, Professor do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro



o júri

presidente

Professor Doutor José Manuel Matos Moreira
Professor Auxiliar, Universidade de Aveiro

vogais

Professor Doutor Carlos Manuel das Neves Santos
Professor Auxiliar, Universidade de Aveiro

Professor Doutor Diogo Nuno Pereira Gomes
Professor Auxiliar, Universidade de Aveiro

agradecimentos

Ao Professor Diogo Gomes pelo acompanhamento que me ofereceu enquanto professor e orientador e por toda a ajuda que me deu ao longo destes meses de desenvolvimento. Sempre me fascinou os temas relacionados com IOT e sem dúvida que ele foi a escolha mais acertada ao escolher este tema para a dissertação.

Aos meus pais e irmã pelo apoio que me deram ao longo dos 5 anos que passei. Agradeço imenso aos meus pais porque sem eles nada disto seria possível. À minha irmã por todas as vezes que me ouviu a desesperar com um erro fácil de resolver, mas que a incomodava e não a deixava estudar.

A todos os meus colegas e amigos que ao longo destes 5 anos me proporcionaram experiências académicas inesquecíveis. Ligações que duraram durante anos e que certamente não vão deixar de existir.

À minha restante família por sempre ter estado lá para mim e me ter dado força e incentivo quando a preguiça falava mais alto.

Palavras-chave Casas inteligentes, IOT, automação

Resumo

O recurso a automações surgiu com o objetivo de facilitar a vida aos utilizadores que pretendam tornar a sua habitação numa habitação mais inteligente e autónoma mas, devido à dificuldade de certos utilizadores ao nível da administração de sistemas, habitualmente recorrem a plataformas externas para a criação de automações da forma mais simplificada.

Nesta dissertação é proposta a criação de um editor que corrigisse as falhas apresentadas pelas atuais plataformas oferecidas pelo mercado. Utilizando ferramentas que possibilitassem o cumprimentos dos requisitos, criou-se o Whenite, uma plataforma que ajudasse os utilizadores a criar as suas automações. Foi também feito um estudo sobre plataformas de tratamento de erros e configurada no Whenite, de modo a possibilitar a resolução de erros.

Aplicando testes de usabilidade à plataforma e tendo em conta feedback obtido, constatou-se que a mesma foi bem aceite pela comunidade.

Keywords Smart houses, IOT, automations

Abstract

The use of automations was created to make life easier for users who want to make their home more intelligent and autonomous but, due to the difficulty of certain users at the level of system administration, they usually use external platforms to create automations in the most simplified way.

In this Masters' Thesis it's proposed the creation of an editor to correct the flaws presented by the current platforms offered by the market. Using tools that made it possible to meet the requirements, Whenite was created, a platform that helped users to create their automations. A study on error handling platforms was also carried out and configured in Whenite to enable error resolution.

Applying usability tests to the platform and taking into consideration feedback obtained, it was found that it was well accepted by the community.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Desde o ínicio dos tempos, o Homem sempre procurou abrigo e proteção nas cavernas, locais estes que mais tarde se converteriam nas atuais habitações [1]. Ao longo dos anos, o Homem tem vindo a procurar aperfeiçoar as suas tarefas diárias de modo a consumir menos tempo e a obter uma sensação de maior conforto [2] e, favorecido pelas novidades e evolução da tecnologia ao nível das contruções, foi possível atender a essas necessidades com a criação dos chamados "edifícios inteligentes".

Com o aparecimento destes "edifícios inteligentes", foi dada a capacidade aos utilizadores de criarem regras de forma a automatizá-la, oferecendo aos mesmos uma sensação de conforto que não seria possível se essas tarefas tivessem de ser realizadas manualmente.

Ao longo deste capítulo é introduzido o conceito de casa inteligente e posteriormente são apresentadas soluções que surgiram para ajudar a automatizar as habitações, os assistentes. É apresentado uma explicação do funcionamento e arquitetura de cada assistente bem como apresentadas as interfaces que cada um dos assistentes oferece para criar automações.

1.1 Casas Inteligentes

1.1.1 Introdução ao Conceito

As casas inteliguentes resultam de uma tecnologia ou uma junção de tecnologias recentes que permitem a gestão de todos os recursos habitacionais, tornando assim as habitações em edifícios inteligentes [3].

Este novo conceito tem vindo a captar cada vez mais a atenção das pessoas, também por ser uma área que despoleta bastante interesse. Está associado ao controlo e automação das habitações, tendo como objetivos principais providenciar aos habitantes da habitação um maior conforto mas também uma maior segurança, seja a nível de deteção de situações de emergência, como incêndios, ou a nível da deteção e sinalização de intrusões por parte de terceiros, por exemplo assaltos [4].

A palavra domótica é um termo frequentemente usado de maneira vaga para definir qualquer tipo de automação residencial e uso de dispositivos inteligentes, sendo que a domótica é vista como uma parte de um termo mais abrangente, as Casas Inteligentes [5].

No que diz respeito ao conforto oferecido pelo aparecimento das casas inteligentes, as possibilidades são imensas, desde o controlar da iluminação à programação do ligar/desligar dos vários equipamentos com horários específicos [4].

As casas inteligentes podem ser uma solução bastante interessante para pessoas com deficiências ou para pessoas idosas, melhorando assim a sua qualidade de vida e autonomia. Um sistema destes pode ser visto como um "mordomo"para este tipo de pessoas, pois as auxilia a realizar as tarefas domésticas [1].

1.1.2 Evolução

Como foi dito anteriormente, as casas inteligentes surgiram da evolução tecnológica associada à Domótica e foi possível juntar aos automatismos que o predecessor oferecia a inteligência e capacidade de realizar as tarefas de casa de forma inteligente, sem ser preciso o utilizador despoletar uma ação. A ideia base destas tecnologias é a presença pervasiva de vários componentes à nossa volta (sensores, smartphones, entre outros) que, através da comunicação e interação entre eles, permitem alcançar certos objetivos [6][7].

O termo casa inteligente começou quando foi criado o primeiro controlo remoto, o que permitiu aos utilizadores não terem de se deslocar até junto da máquina para a controlar.

Em 1966 surgiu o primeiro assistente para automações inteligentes, o Echo IV. Este dispositivo permitia às pessoas da casa criar listas de compras, regular a temperatura da casa, bem como ligar/desligar dispositivos [8]. A criação dos micro-controladores em 1971 resultou numa diminuição dos preços dos dispositivos eletrónicos, tornando assim a tecnologia mais acessível [8].

Os anos 2000 trouxeram um aumento da popularidade das casas inteligentes [9]. Várias tecnologias surgiram e foram lentamente integradas na automação residencial e, apartir desse momento, o utilizador passou a ter o poder de, por exemplo, controlar o aquecimento, as luzes, ou o alarme de casa.

Com a inserção de sensores nas casas, é também possível saber se alguém está em casa ou então se a humidade do jardim está nos valores corretos, caso contrário, é enviado um comando ao controlador da rega do jardim para ligar automaticamente.

É aqui que entra a inteligência e a passagem do termo domótica para casa inteligente. A inteligência analisa os dados provenientes dos sensores que temos espalhados pela habitação e ela própria executa o que for preciso sem ser necessário a interação do utilizador, pois a habitação é suficientemente inteligente para executar as ações sem intervenção do utilizador.

1.1.3 Arquiteturas

No que diz respeito às arquiteturas dos sistemas de automação residenciais, estas podem ser divididas em dois grandes grupos, as arquiteturas centralizadas e as arquiteturas descentralizadas. [1]

- **Centralizadas** - Nesta arquitetura existe uma unidade central à qual todos os dispositivos se encontram ligados. Esta central, ou controlador, tem como objetivo receber toda a informação proveniente dos sensores e, após a decisão do que terá de ser realizado, trasmitir os comandos aos atuadores com a finalidade de desempenhar a operação pretendida.[1] Na figura 1.1, baseada em [10], temos esquematizada uma arquitetura centralizada.

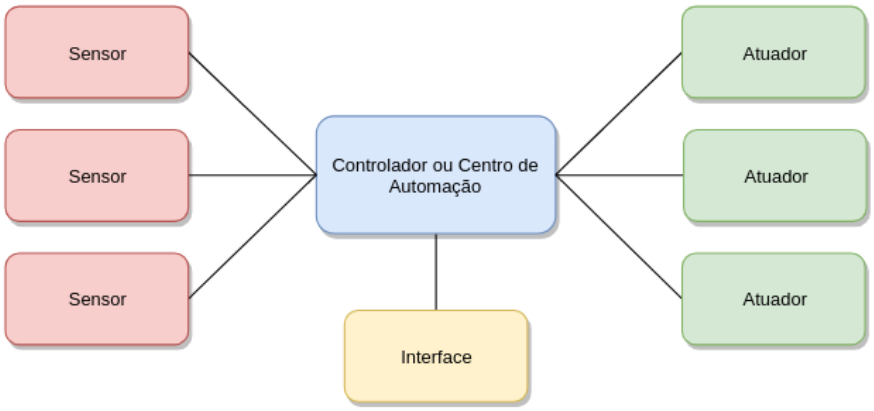


Figura 1.1: Esquema de uma arquitetura centralizada [10]

- **Descentralizadas** - Neste tipo de arquiteturas existem vários tipos de elementos com processamento próprio, sendo que têm um papel específico dentro do sistema. Estes elementos são interligados por uma rede e comunicam entre si enviando e recebendo dados dos sensores e dos atuadores [2]. Na figura 1.2, baseada em [10], temos esquematizada uma arquitetura descentralizada.

1.1.4 Aplicações

A automação está cada vez mais em amplo crescimento e cada vez mais são estudadas alternativas para que surjam novas técnicas para aprimorar os sistemas de automação residencial e também para que surjam novos produtos com este mesmo fim [2].

Podemos agregar os sistemas de automação em quatro grandes áreas, sendo elas **Segurança**, **Conforto**, **Gestão de Recursos** e **Comunicação** [11].

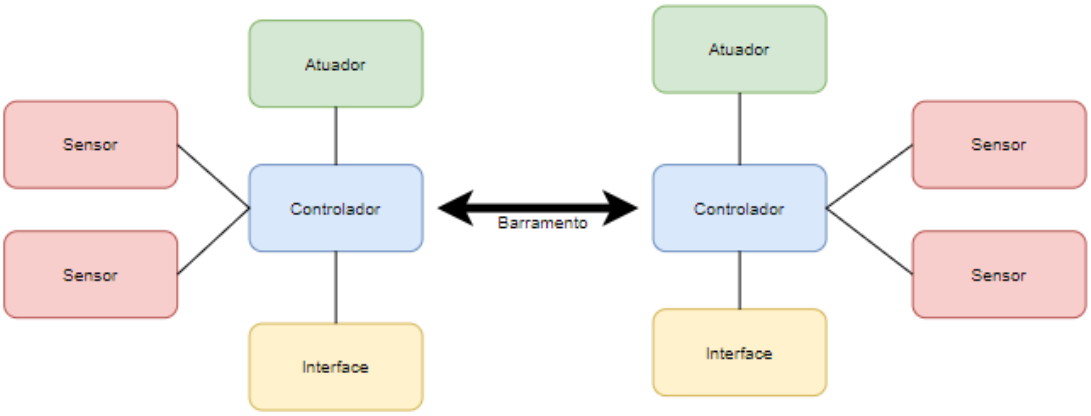


Figura 1.2: Esquema de uma arquitetura descentralizada [10]

- **Segurança**: Ao integrar diversos sistemas em torno de um único sistema, as casas inteligentes possibilitam um aumento nos padrões de segurança, através da utilização de todas as potencialidades dos diversos sistemas envolvidos [12].

  Esta área das casas inteligentes proporciona uma proteção contra intrusões, avarias e incidentes, com recurso a alarmes, câmaras de vigilância e alarmes técnicos de incêndios, inundações, fugas de gás, entre outros [13]. Assim que o sistema detetar alguma pessoa a entrar na habitação apartir de certo horário, pode imediatamente despoletar uma ação por exemplo acendendo as luzes e ligando uma sirene.

- **Conforto**: Estes sistemas, tendo como principal objectivo oferecer o maior conforto aos utilizadores da habitação, são responsavéis então por providenciar aos mesmos diversas funcionalidades sem que seja necessário que o utilizador se desloque ou tenha de fazer ações demoradas.

  Tendo como desplotadores das ações os sensores, é útil abrir os estores ou acender as luzes quando existir alguém na divisão, caso contrário não existe necessidade de as luzes se encontrarem acesas por exemplo. Se os utilizadores poderem acender as luzes ou ligar o aquecimento mesmo estando sentados no sofá através da interação com um controlador do sistema, estamos a oferecer um maior conforto a esses utiizadores.

- **Gestão de Recursos**: Segundo Alves e Mota [12] a utilização correta da energia não implica a ausência do consumo, mas sim a racionalização do mesmo. Relacionado com o que foi falado anteriormente, o facto de ser possível controlar os sistemas de iluminação/aquecimento entre outros, podemos apenas proceder à ativação dos mesmos apenas quando necessário, o que oferece uma maior racionalização dos recursos energéticos da habitação, e consequentemente um alívio financeiro.

- **Comunicação**: Esta área abrange as trocas de informação e recursos entre os diversos

dispositivos presentes na habitação.

Um sistema destes é composto por uma rede de comunicação entre uma série de dispositivos, com o objetivo de recolher informação e atuar sobre o ambiente residencial de forma a exercer o controlo e supervisão da casa [13]. Os meios de comunicação destas redes podem ser vários, havendo a possibilidade de recorrer a tecnologia sem fios (radiofrequência e infravermelhos) ou então a comunicação por cabo (rede elétrica, cabos de rede ou então fibra ótica).

As casas inteligentes podem ter diversas funcionalidades, dependendo daquilo que o utilizador pretende, podendo este optar por uma instalação mais básica, normalmente relacionada com a segurança e o conforto, ou então um sistema mais completo que alberga todas as funcionalidades e potencialidades que estas casas podem oferecer.

Capítulo 2

Estado da Arte

Ao longo deste capítulo é apresentado um levantamento do estado da arte acerca de plataformas externas aos seus assistentes às quais os utilizadores recorrem para terem uma maior facilidade na criação das suas automações.

Para além disso, é feito o levantamento do estado da arte de algumas plataformas que são utilizadas para efetuar a monitorização de erros em aplicações de diversos tipos. Estas plataformas permitirão uma maior facilidade no controlo dos erros do produto que terá como finalidade colmatar os problemas associados aos atuais métodos para criação de automações.

2.1 Assistentes

A domótica está relacionada com o ato de automatizar e, ligada à inteligência que a evolução da tecnologia ofereceu, foi possível tornar automáticas diversas tarefas que um habitante comum teria de fazer manualmente em sua casa. Posto isto, podemos concluir que uma automação é um processo que tem como objetivo executar um conjunto de instruções com a mínima interação do utilizador possível. No que diz respeito aos vários tipos de automação que podem existir, podemos separar as automações em dois grupos: as **automações controladas pelo utilizador** e as **automações baseadas em regras** [14].

Nas **automações controladas pelo utilizador**, o habitante realiza explicitamente uma ação que leva a que vários acontecimentos aconteçam devido a essa ação, por exemplo, um botão (virtual ou físico) que tem como função desligar todas as luzes de casa [14].

Nas **automações baseadas em regras**, as ações são desplotadas através de variados eventos, podendo estar associados a sensores ou simplesmente a uma certa altura do dia. Exemplos destas automações podem ir desde acender as luzes do corredor quando uma pessoa caminha pelo mesmo ou então temos como exemplo baixar os estores e acender as luzes exteriores ao fim do dia, quando chega a hora do pôr do sol [14].

Com a necessidade de permitir que os habitantes pudessem gerir a sua habitação de forma inteligente começaram a ser desenvolvidas plataformas comerciais que o utilizador poderia

adquirir e utilizar de forma a ser mais fácil realizar a automação da sua habitação como por exemplo a Amazon Alexa e o Google Home. Com o lançamento destas plataformas surgiram também soluções Open-Source, como é o caso Home-Assistant, openHAB e Domoticz.

2.1.1 Amazon Alexa

A Amazon Alexa é um assistente controlado por voz desenvolvido pela Amazon para alguns dos seus dispositivos, nomeadamente o Amazon Echo, Echo Dot e mais recentemente o Echo Show [15][16][17].

Os dispositivos na qual este assistente está inserido, devido ao facto de se encontrarem ligados à cloud, têm também a capacidade de controlar a habitação, uma vez que conseguem comunicar com dispositivos de IOT que são compatíveis com este assistente [17].

Como funciona?

Como referido acima, os dispositivos onde a Amazon Alexa está incorporada estão conectados à Cloud. Esta ligação permite que os controlos ditos pelo utilizador sofram um tratamento e seja convertido em comandos. Esses comandos são executados por serviços que se encontram também ligados à Cloud. Após o processamento do pedido efetuado pelo utilizador, a Cloud faz com que o utilizador seja avisado através de comandos executados pelos disposistivos.

Na figura 2.1 podemos observar um esquema exemplificativo do modo de funcionamento deste assistente [18].

Figura 2.1: Como funciona a Amazon Alexa [18]

O utilizador fala diretamente com o dispositivos que comunica o conteúdo falado com a Cloud. Esta é responsável pelo processamento do pedido e extração de comandos que são executados através de serviços que também se encontram conectados com a Cloud.

Automação

As automações na Amazon Alexa são conhecidas por outro nome, rotinas. Estas rotinas são constituídas, à semelhança dos assistentes anteriormente apresentados, por *triggers*, condições e ações. Como este assistente é controlado por voz, uma das maneiras de despoletar uma rotina é com uma determinada frase, isto é, quando o utilizador disser "Alexa, vou dormir"são despoletadas um conjunto configurável de ações definidas pelo utilizador [19].

O primeiro passo para a criação de uma rotina com a Amazon Alexa é abrir a aplicação do assistente e escolher criar uma nova rotina [20]. Após este passo é pedido ao utilizador para selecionar o tipo de *trigger* que pretende usar na rotina. Os *triggers* podem ser de diversos tipos mas para este tipo de assistente, o mais comum é o comando de voz.

Podemos observar na figura 2.2 os primeiros passos da criação de uma rotina neste assistente.

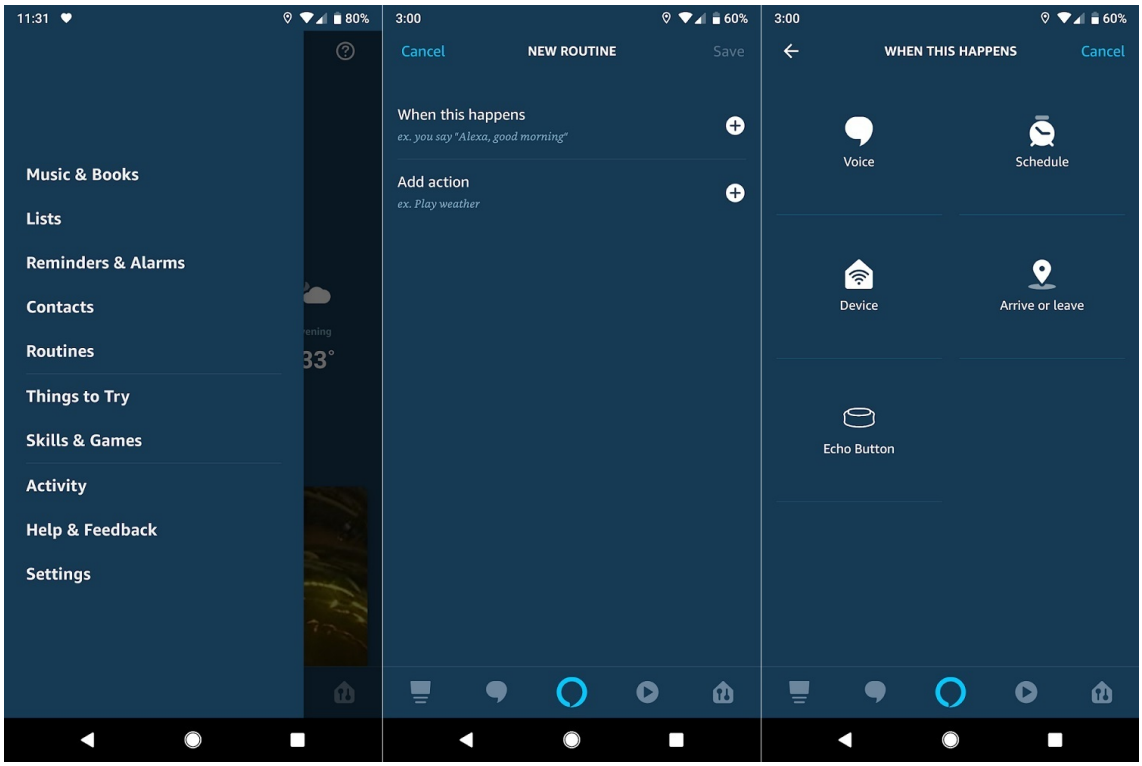


Figura 2.2: Primeiros passos da criação de uma rotina na Amazon Alexa [20]

Após o utilizador escolher os *triggers* da sua rotina, o passo seguinte é indicar o que pretende que aconteça. É possível, nas ações, controlar dispositivos que se encontrem ligados ao assistente. Para o caso mencionado acima do utilizador dizer "Alexa, vou dormir", uma das ações que poderia ser adicionada a esta rotina é o apagar das luzes do quarto, por exemplo.

Na figura 2.3 são apresentadas as interfaces de adição de ações na aplicação da Amazon Alexa.

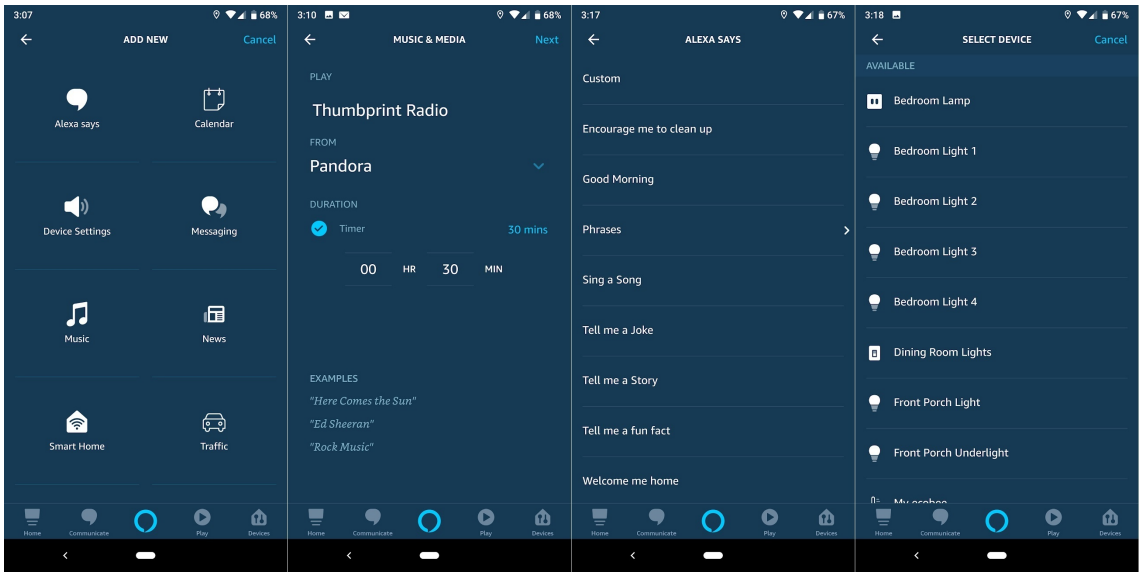


Figura 2.3: Indicação das ações pretendidadas na rotina [20]

2.1.2 Google Home

Estando colocado no mesmo patamar da Amazon Alexa, o Google Home foi lançado em 2016 pela Google. Este assistente oferece também um variado leque de dispositivos compatíveis para que possamos ter o total controlo sobre a nossa habitação. O Google Home tem incorporado nele o Google Assistant, um assistente de voz que mantém contacto através da voz com o utilizador, à semelhança do assistente da Amazon.

Como funciona?

Como foi dito, também o Google Home está conectado a uma Cloud que é responsável por interpretar as falas dos utilizadores e converter em comandos. Casos estejam conectados com o Google Home dispositivos IOT, é possível atuar sobre eles. Na figura 2.4 temos esquematizado resumidamente como funciona o Google Home [21].

Automação

À semelhança de um dos seus principais concorrentes, a Amazon Alexa, também o Google Home funciona através de rotinas. Estas rotinas permitem ao utilizador realizar diversas ações apenas com uma frase [22]. Apesar de já haver rotinas pré configuradas, que o utilizador pode configurar como bem prentender, também tem a liberdade de criar novas rotinas com frases configuradas por si. Além da voz, é possível ativar rotinas através de horários configuráveis.

O primeiro passo para criar uma rotina no Google Home, é aceder ao assistente e selecionar o menu das rotinas, como se pode observar na figura 2.5.

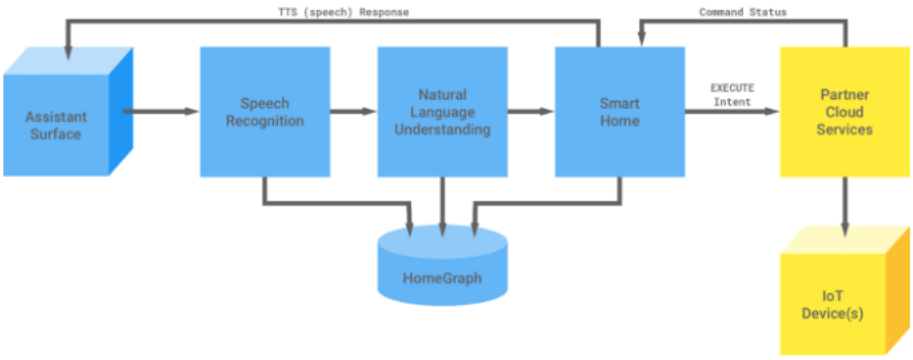


Figura 2.4: Como funciona o Google Home [21]

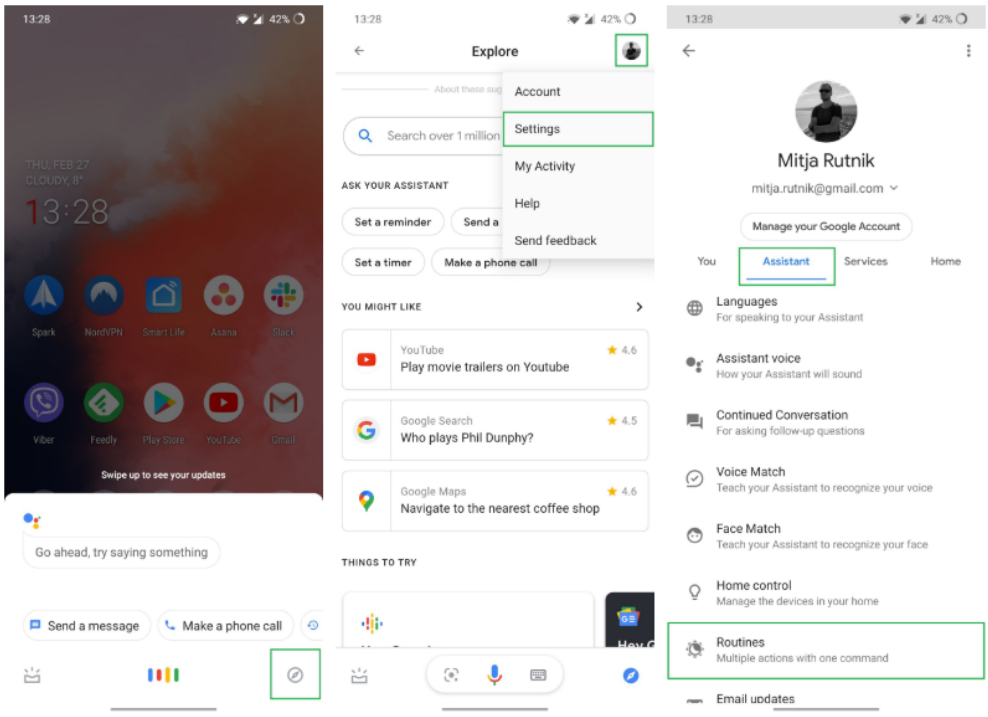


Figura 2.5: Como aceder ao menu das rotinas no Google Home [22]

Após entrar no menu de criação de rotinas, outro menu irá aparecer onde o utilizador seleciona o despoletador das ações e as ações que pretendem que sejam realizadas [22]. Como dito anteriormente, existem duas formas distintas de despoletar uma automação: **frases** ou **horários específicos**.

Na figura 2.6 podemos visualizar o menu no qual é possível configurar a rotina.

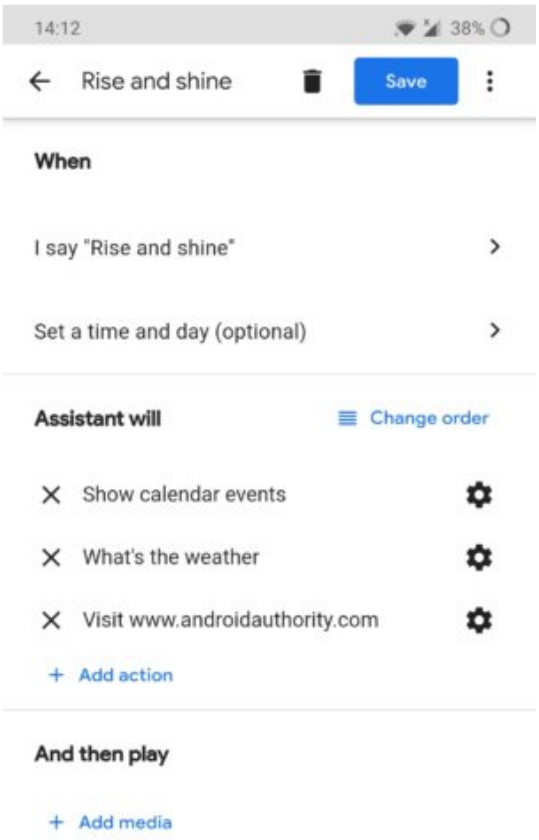


Figura 2.6: Menu de criação de nova rotina no Google Home [22]

2.1.3 Home-Assistant

O Home-Assistant é uma plataforma de automação gratuita, que pode ser facilmente instalada em qualquer máquina que consiga executar Python 3 [2]. Na figura 2.7 podemos observar a interface deste assistente.

Este sistema veio colmatar algumas dificuldades sentidas na atualidade, uma vez que grande parte dos sistemas existentes utilizam soluções proprietárias, ou seja, não era possível a integração com outras marcas. Veio também colmatar dificuldades sentidas ao nível da comunicação entre os dispositivos, uma vez que são utilizados diferentes protocolos de acordo com os dispositivos que são utilizados, como por exemplo Zigbee, Z-Wave, Wi-fi, Radio-Frequência e Infra-Vermelho [24].

O Home Assistant oferece interfaces para diverso sistemas operativos desde Android, IOS ou até mesmo Windows Store, o que possibilita controlar a nossa habitação de onde quer que nos encontremos.

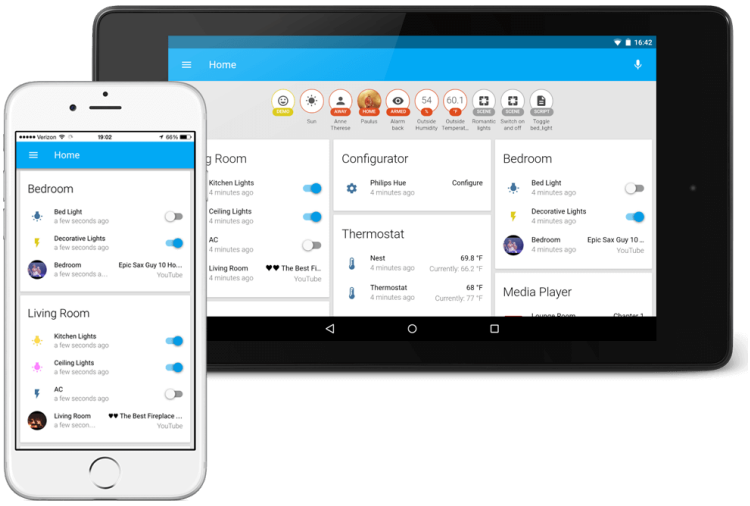


Figura 2.7: Interface da plataforma Home-Assistant [23]

Arquitetura

Na figura 2.8 temos representada a arquitetura do Home Assistant [25]. Podemos destacar duas partes fundamentais nesta arquitetura, sendo elas o bloco "Home Control" e o bloco "Internet Of Things".

- **Home Control**: O bloco denominado de "Home Control" é o componente responsável por reunir informação proveniente dos dispositivos que estiverem conectados ao Home Assistant e também é responsável pelo seu controlo.
- **Internet Of Things**: Este bloco representa todos os dispositivos - sejam eles luzes, interruptores, sensores - que podemos ter conectados ao Home Assistant. Estes dispositivos comunicam com o "Home Control" para indicar informações provenientes dos mesmos ou então para receber comandos com a finalidade de alterar os seus estados.

Para o bloco "Home Control" ter a capacidade de controlar dispositivos e processar informação proveniente desses, existe o chamado "Home Assistant Core"que apresenta quatro blocos que permitem ao Home Control essas funcionalidades, como pode ser observado na figura 2.9:

- **Event Bus**: O "Event Bus"funciona como o coração do Home Assistant, porque ele é o responsável por atentar a eventos ocorridos e pode fazer despoletar outros eventos.
- **State Machine**: Mantém o registo dos estados atuais de todos os componentes e despoleta o evento *state_changed* sempre que um dos dispositivos altera o seu estado.
- **Service Registry**: Mantém-se à escuta de eventos do tipo *call_service* e permite o registo de novos serviços.

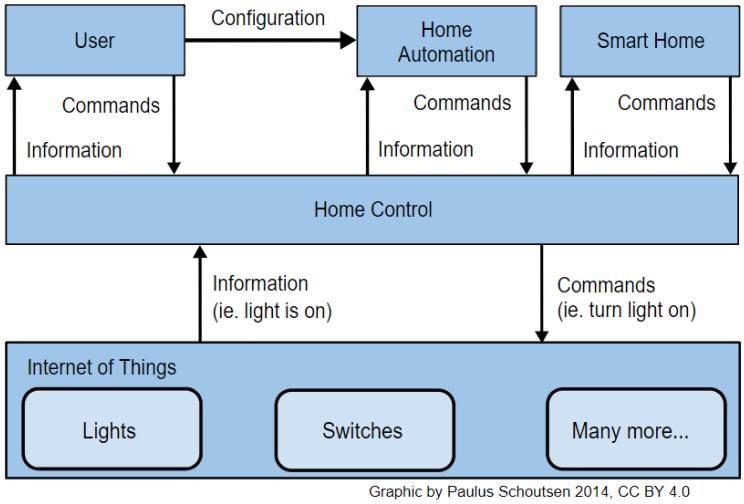


Figura 2.8: Arquitetura do Home Assistantt [25]

- **Timer**: Envia a cada segundo um evento do tipo *time_changed* para o "Event Bus".

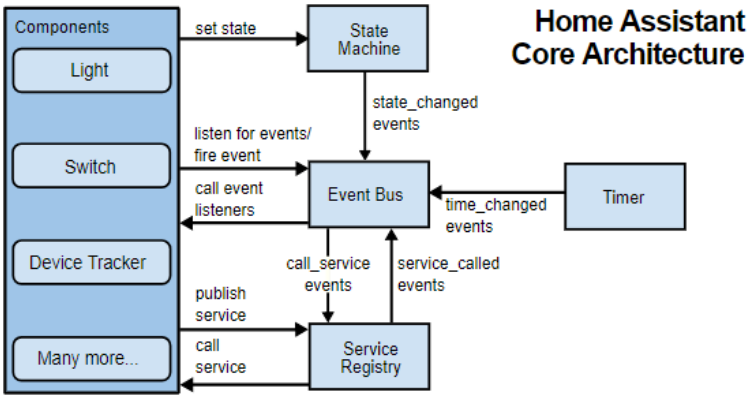


Figura 2.9: Arquitetura do Home Assistant Core [25]

O Home Assistant pode ser complementado com componentes externos onde cada um dos componentes é responsável por um domínio específico do Home Assistant [25]. Estes componentes são escritos em Python e têm a capacidade de reagir a eventos, disponibilizar serviços e possuir estados. O próprio Home Assistant oferece uma grande quantidade de integrações disponíveis [26].

Tomando em consideração o que foi dito anteriormente podemos então estabelecer uma complementaridade entre a arquitetura do Home Assistant e os componentes externos que possam vir a ser adicionados, resultando uma arquitetura mais completa como é mostrada na figura 2.10 [25].

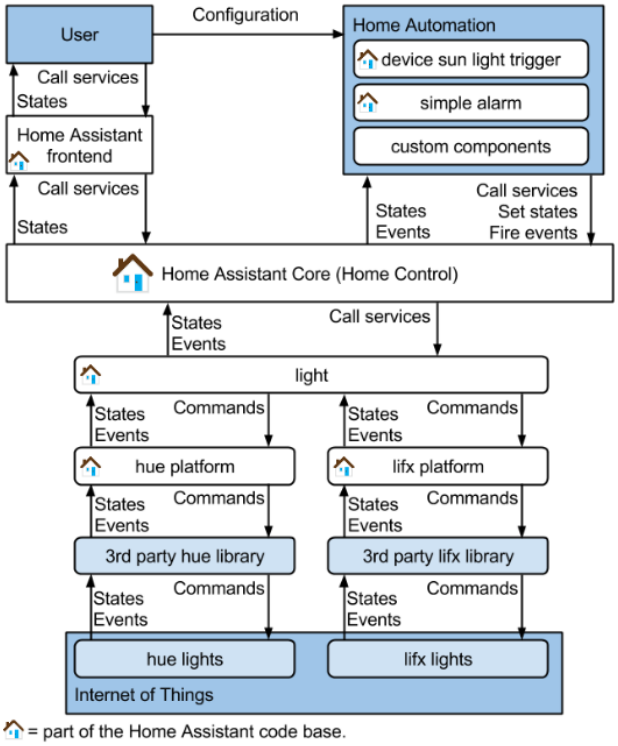


Figura 2.10: Arquitetura completa do Home Assistant [25]

Automação

O Home Assistant oferece uma plataforma inerente ao assistente para possibilitar a criação de uma automação. Estas automações dividem-se em três partes: os *triggers*, as condições, e as ações. Os *triggers* representam os eventos que têm que acontecer para que uma ação seja executada. As condições são, como o próprio nome indica, condições que têm de se verificar para que a ação seja executada. As ações representam o que queremos que aconteça quando o *trigger* acontece.

Se pretendermos criar uma automação que ligue o interruptor da cozinha quando ocorrer o pôr do sol, conseguimos imediatamente estabelecer a correspondência entre essa automação e as partes constituintes da automação. O *trigger* será o pôr do sol e a ação ligar o interruptor da cozinha. Na figura 2.11 temos apresentado a interface de criação de automações do Home Assistant.

As automações no Home Assistant estão construídas no formato YAML e todas as automações criadas encontram-se no ficheiro *automations.yaml* localizado no diretório do Home Assistant, diretório este que também contém os ficheiros de configuração do Home Assistant.

Para a automação referida acima, o código YAML correspondente é:

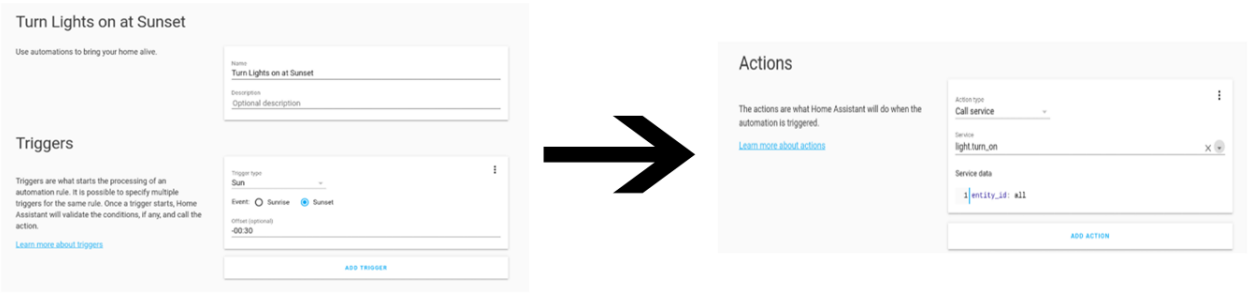


Figura 2.11: Interface para criar automações no Home Assistant. Os *triggers* à esquerda e as ações à direita [25]

```
- id: '1589318685862'
  alias: Ligar Luz da Cozinha ao anoitecer
  description: ''
  trigger:
  - event: sunset
    platform: sun
  condition: []
  action:
  - data: {}
    entity_id: switch.sonoff_kitchen
    service: switch.turn_on
```

Ao analisar o bloco de código apresentado acima, observamos que cada automação tem um id aleatório de modo a não existir sobreposição de automações. Como nesta automação não inserimos nenhuma condição, este componente na automação encontra-se vazio, representado com [].

2.1.4 openHAB

À semelhança do Home Assistant, outra plataforma Open-Source que surgiu para oferecer aos utilizadores a possibilidade de automatizar a sua habitação foi o *Open Home Automation Bus*, também conhecido como openHAB [27]. É um controlador implementado em Java e que é capaz de correr em várias plataformas como Linux, Windows e MacOSx. Para além destas plataformas, o openHAB também é capaz de correr em RaspberryPi, Docker entre outros [28]. Na figura 2.12 podemos observar a interface deste assistente.

O openHAB oferece uma integração com inúmeros dispositivos e oferece uma interface uniforme que reduz a dependência de subsistemas que possam ser envolvidos. Esta plataforma também oferece aos utilizadores a possibilidade de criar as suas automações na própria plataforma.

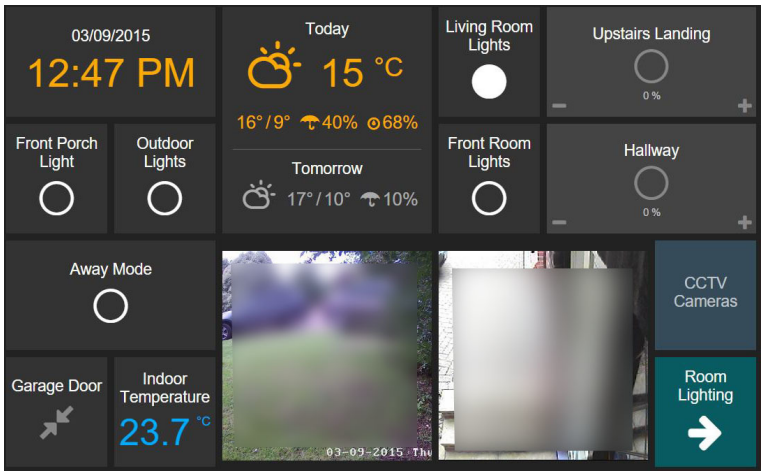


Figura 2.12: Interface do openHAB [29]

A figura 2.13 ilusta a relação entre *things*, *channels* e *items*, que são os conceitos básicos do openHAB. [30]

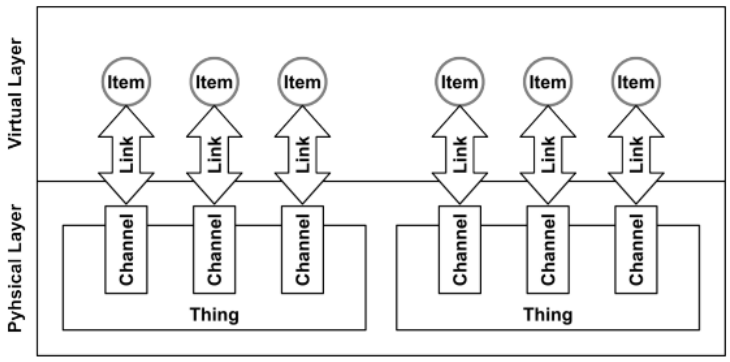


Figura 2.13: Camada virtual da plataforma openHAB [30]

Os componentes denominados de *things* são dispositivos ou serviços físicos que podem ser ligados ao sistema. Os *Items* são uma representação gerada pelo openHAB de informação proveniente dos dispositivos. Os *Channels* são o "elo de ligação lógica"[27] entre os dois componentes referidos anteriormente.

Arquitetura

O openHAB é acente numa framework OSGi onde o Eclipe Equinox [31] e o Apache Karaf [32] são utilizados como ambiente de desenvolvimento [30]. Por ser baseada em OSGi, oferece uma arquitetura bastante modular, o que permite adicionar e remover funcionalidades em tempo real sem necessidade de parar o serviço [33]. Na figura 2.14 podemos observar a arquitetura do openHAB.

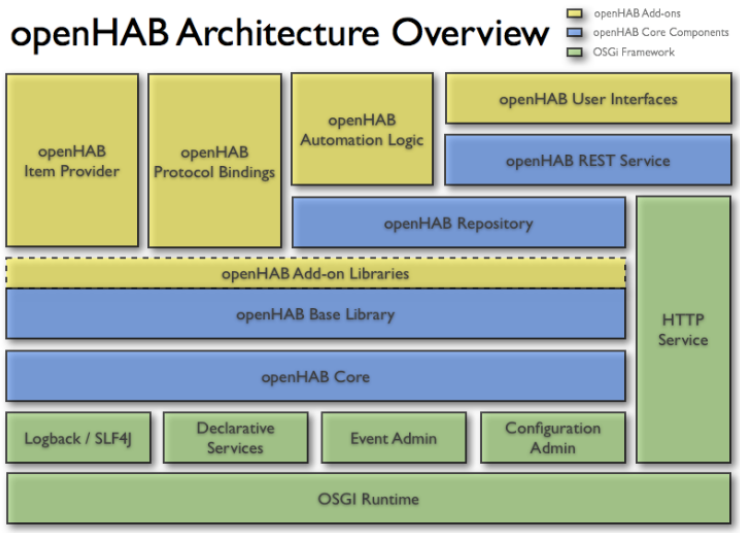


Figura 2.14: Arquitetura do openHAB [28]

Comunicação

No que diz respeito a comunicação, podemos identificar dois canais de comunicação internos no openHAB:

1. Um *Event Bus* assíncrono
2. Um repositório a que se pode recorrer

O *Event Bus* é o serviço base do openHAB. Todos os componentes devem utilizar este serviço para informar os restantes de eventuais eventos que possam surgir ou recorrer a ele para atualizar o seu estado de acordo com eventos ocorridos. É muito semelhante ao *Event Bus* que foi explicado no Home Assistant.

O Repositório faz o tracking do atual estado de todos os dispositivos. Se apenas for necessário obter o atual estado de um dispositivo, o openHAB recorre ao repositório para devolver esse estado e não solicita ao *Event Bus*. Outra vantagem do repositório deve-se à persistência dos estados, isto é, no caso de um sistema precisar de ser reiniciado, todos os estados em que se encontravam os dispositivos são retomados.

A figura 2.15 demonstra como esses canais de comunicação são utilizados [33].

Automação

No que diz respeito a automações no openHAB, estas têm o nome de *Rules*, à semelhança do Home Assistant, são do tipo *Trigger - Action*, isto é, dado um *triggers* despoleta uma ou várias ações.

Para definir uma *Rule* existe um formato específico que se tem de respeitar para o openHAB interpretar a automação. Podemos ter as automações que quisermos desde que o nome

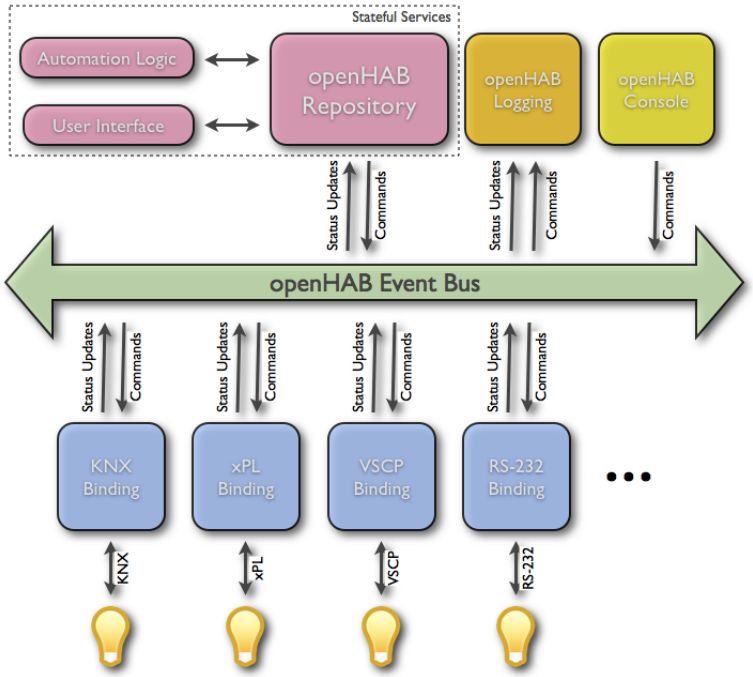


Figura 2.15: Canais de Comunicação do openHAB [34]

---

do ficheiro seja do formato '*.rules' e esse ficheiro esteja contido no diretório dedicado às automações '$OPENHAB_CONF/rules/'.

O formato de um ficheiro de automação do openHAB é o seguinte: [27]

```
rule "<RULE_NAME>"
when
    <TRIGGER_CONDITION> or
    <TRIGGER_CONDITION2> or
    ...
then
    <SCRIPT_BLOCK>
end
```

- <**RULE_NAME**>: Cada *Rule*/Automação tem de ter um nome único. É recomendado que se escolha um nome que faça sentido quando falado;
- <**TRIGGER_CONDITION**>: À semelhança dos *triggers* do Home Asssitant, esta condição determina quando a automação é despoletada. Podem ser utilizados vários *triggers* utilizando a *keyword* 'or';
- <**SCRIPT_BLOCK**>: Contém o código que deverá ser executado quando se verificar uma <TRIGGER_CONDITION>.

Tendo em conta o formato do ficheiro que foi apresentado acima podemos, por exemplo, criar uma automação que ligue a tomada da TV quando deteta a presença de um habitante.

```
rule "Turn on Wall TV Plug"
when
  Item Presence_Mobile changed from OFF to ON
then
  Wall_TV_Plug.sendCommand(ON)
end
```

2.1.5 Domoticz

O Domoticz [35] é outro dos assistentes que permite aos utilizadores o controlo de dispositivos inteligentes na sua habitação. Na figura 2.16 retirada o website do Domoticz, temos apresentada a interface de controlo deste assistente, que também permite ao utilizado a utilização de temas.

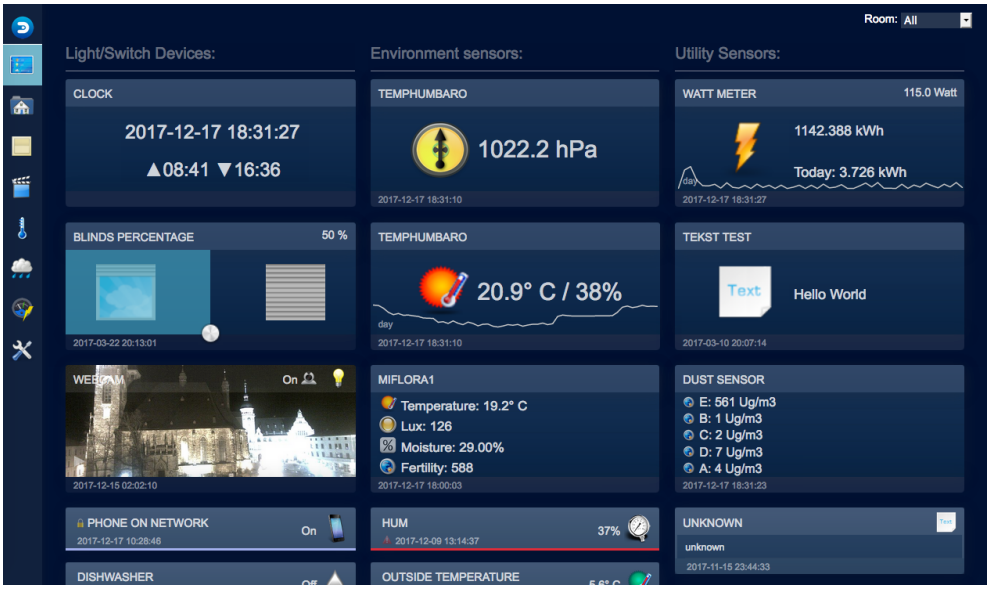


Figura 2.16: Interface do Domoticz [35]

É totalmente gratuito e é compatível com inúmeros sistemas operativos. A sua interface é criada em HTML5 juntamente com Bootstrap o que o torna responsivo para qualquer ambiente, seja ele mobile ou web.

Este assistente oferece compatibilidade com inúmeras tecnologias de comunicação, o que permitirá ao utilizador ligar qualquer tipo de dispositivo inteligente.

Automação

As automações no Domoticz são definidas através de código, podendo ele ser escrito em PHP ou LUA, o que permite automatizar como bem entendermos[36]. Além de permitir a escrita de scripts, oferece também ao utilizador a possibilidade de utilizar um editor visual, o Blockly [37] que permite uma escrita de código de forma visual.

O Blockly foi desenvolvido pela Google e tem como objectivo a programação de forma visual. É utilizado essencialmente para criação de código. Apresenta uma barra lateral com todos os componentes necessários para a criação de código onde o utilizador apenas tem de selecionar os componentes que prentende e conectar uns com os outros.

Através desta programação visual, o utilizador não precisa de ter conhecimentos ao nível da escrita de código para realizar uma automação. Nas figuras 2.17 e 2.18 podemos visualizar as duas maneiras distitas de criar uma automação no Domoticz.

```
commandArray = {}

if (devicechanged[pulse_counter]) then
   pulses = tonumber(otherdevices_svalues[pulse_counter])
   prev_meter = tonumber(otherdevices_svalues[meter_name])

   base = tonumber(uservariables[meter_base])
   if (base + pulses < prev_meter) then
     -- Pulse counter seems to have been reset.
     print('Pulse count reset; updating base from ' .. base .. ' to ' .. prev_meter)
     base = prev_meter
     commandArray['Variable:' .. meter_base] = tostring(base)
   end
-- print('Meter reading ' .. base .. ' + ' .. pulses .. ' = ' .. base + pulses)
   commandArray['UpdateDevice'] = meter_id .. "|0|" .. base + pulses
end

return commandArray
```

Figura 2.17: Script em LUA do Domoticz [38]

2.1.6 Problema

Como podemos constatar, as automações vêm simplificar a vida do utilizador na realização de tarefas de forma automática dentro da sua habitação.

Apesar das plataformas apresentadas oferecerem uma interface para o utilizador criar automações, esta interface torna-se um pouco confusa quando temos a intenção de fazer uma automação mais complexa. Quando se pretende a realização de muitas ações aquando a realização da automação, a leitura e perceção do fluxo que está a ser criado pode ser díficil.

Para combater essa pequena lacuna, muitos utilizadores optam pela utilização de plataformas externas que oferecem uma interface mais *user-friendly* para criar as suas automações desejadas, plataformas estas que permitem uma leitura mais facilitada da automação criada.

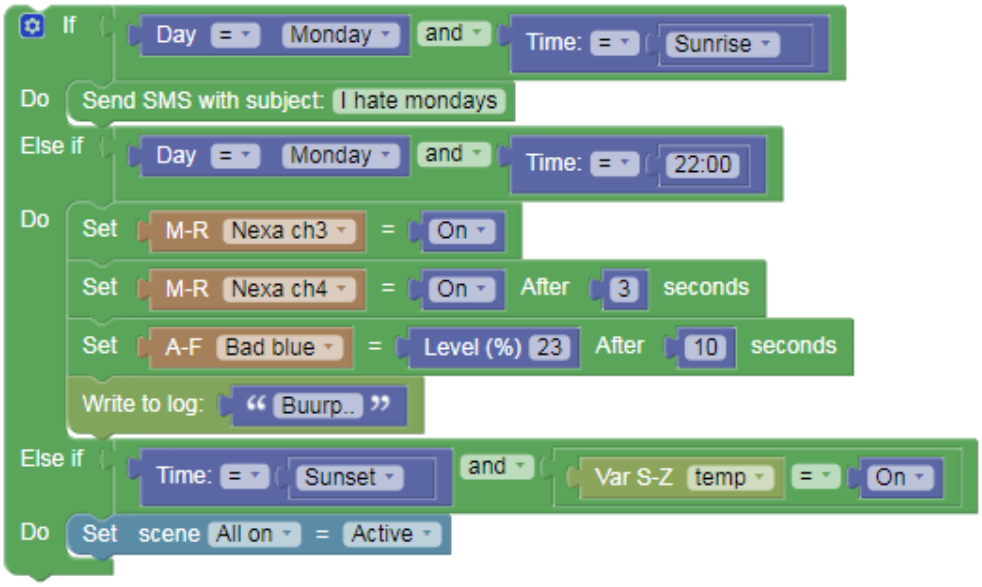


Figura 2.18: Exemplo de script no Domoticz utilizando o Blockly [38]

2.2 Interfaces para Criação de Regras de Automação

No subcapítulo anterior, foram apresentados diferentes assistentes que ajudam a tornar as habitações comuns em habitações mais inteligentes, oferecendo controlo sobre os elementos da mesma e possibilitando a criação de automações de diversas maneiras.

Tendo em conta as diversas formas que possibilitam a criação de automações, este subcapítulo é responsável por apresentar diversas interfaces que podem ser usadas para a criação de regras de automação.

2.2.1 IFTTT - If This, Then That

O IFTTT [39] é um serviço gratuito que pode ser usado para automatizar quase tudo o que um utilizador necessita. Este serviço oferece a possibilidade de conectar uma inúmera quantidade de serviços de variados tipos e utiliza-os para automatizar ações [40]. Por exemplo, se o utilizador tiver o sistema de iluminação da Philips Hue ou outra iluminação inteligente, pode acender uma luz sempre que receber uma mensagem no Facebook.

Na figura 2.19 podemos observar a interface do IFTTT, com algumas Applets já criadas.

Os applets representam o fluxo de instruções que criam uma automação. Este componente do IFTTT é constituído por *triggers* e ações, onde o *trigger* indica quando é que o applet pode ser iniciado e a ação é o resultado da execução do applet.

Além de inúmero applets que já vêm configurados por defeito no serviço, também oferece aos utilizadores a possibilidade de criar os seus próprios applets, apenas tem de escolher o acontecimento que desplota a execução do applet que está a criar e escolher o serviço que é

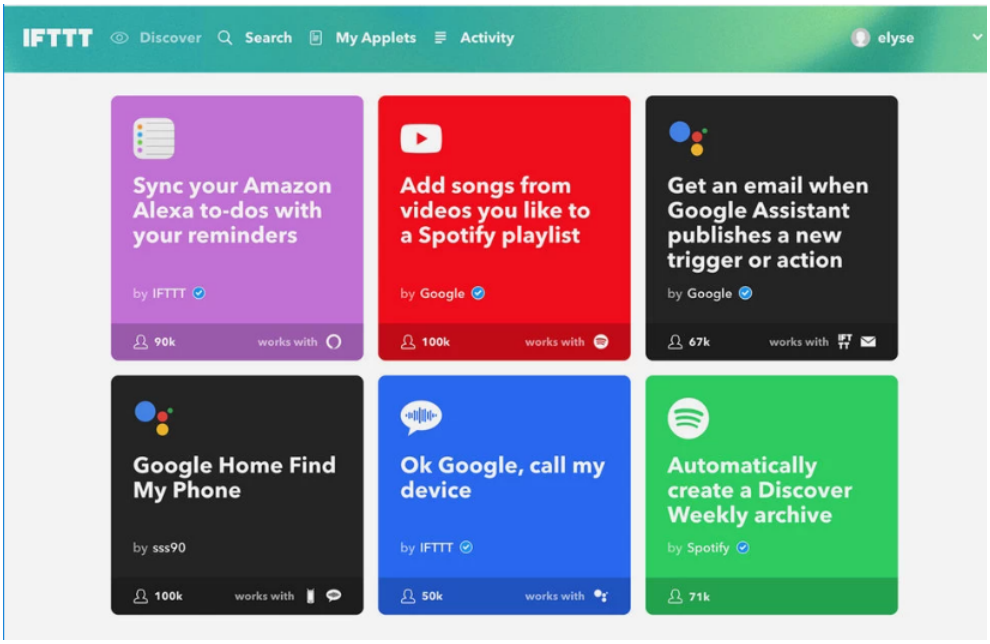


Figura 2.19: Interface do serviço IFTTT [40]

executado posteriormente.

Além dos applets, o IFTTT também pode ser utilizado através de widgets. [39] Estes componentes são utilizados para possibilitar ao utlilizador a execução de um applet com apenas um toque. Estes widgets requerem a instalação da aplicação do IFTTT nos seus dispositivos.

Na figura 2.20 retirada da página do IFTTT, podemos ver os menus onde podem ser utilizados os widgets.

2.2.2 Motores de Regras

A ideia principal associada a motores de regras está relacionada com exteriorizar as lógicas de negócio e aplicação. Um motor de regras pode ser visto como um interpretador sofisticado de *if-then statements* [41].

Uma regra é composta por duas partes essenciais: uma **condição** e uma **ação**. Sempre que a condição é reconhecida, a ação associada é despoletada.

Um motor de regras é uma ferramenta bastante boa para tomadas de decisão uma vez que consegue tomar decisões através de uma grande quantidade de dados de forma rápida e com elevada confiança [41][42][43]. Estes sistemas podem ser usados para criar regras relacionadas com inúmeras situações distintas. Foi proposto por Nhan Bao [44] um termóstato inteligente que regula a temperatura de forma inteligente e apropriada dependendo de situações dinâmicas através de um motor de regras.

Relacionado com a *Internet Of Things*, se pretendermos adquirir dados de vários sensores

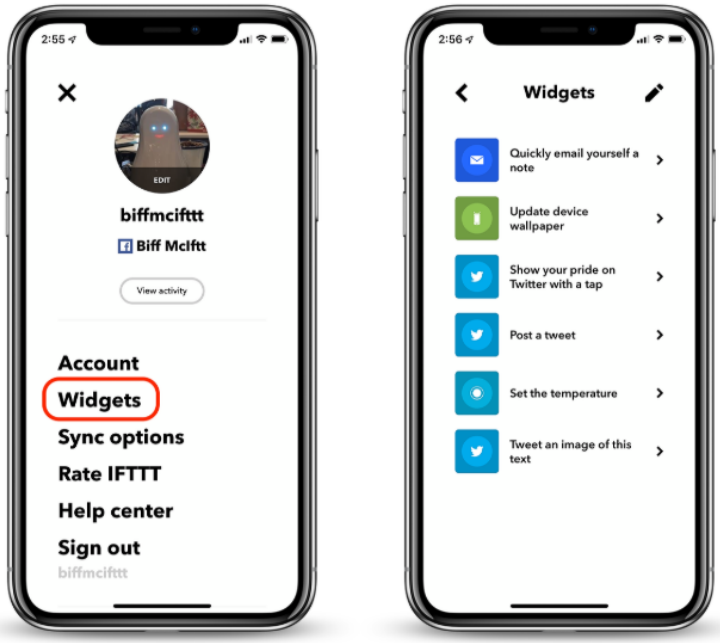


Figura 2.20: Menu de utilização de widgets do IFTTT [39]

e quisermos relacionar diversas ações de acordo com os dados que são recolhidos, um motor de regras é uma ótima solução [43].

Tipos de Motores de Regras

Existem vários tipos de motores de regras que podem ser utilizados, cada um com as suas vantagens de acordo com a finalidade que se pretende [45]. Dentro das várias alternativas existentes, vão ser agora explicados dois tipos de motores de regras: **Motores de Condição-Ação** e **Motores com programação baseada em fluxos**.

Os motores de condição-ação dão resposta aos já referidos *if-then statements*, mas têm uma limitação, uma vez que não permitem várias condições. Esta limitação torna-os por um lado muito restritos na sua capacidade de expressão lógica, mas por outro torna-os muito mais escaláveis [45].

Nos motores com programação baseada em fluxos, os componentes são representados por nós, cada um com um propósitos específico. A ligar esses componentes são efetuadas conexões de modo a criar um fluxo de execução. O Node-Red é um motor de regras que segue esta programação baseada em fluxos [45][46].

2.2.3 Controlo por voz

Este tipo de controlo é muito comum nos assistentes por voz, como é o caso do Google Assistant e da Alexa, que foram mencionados anteriormente.

Apesar de estes assistentes utilizarem a voz fundamentalmente como *trigger* para efetuar um conjunto de ações, existem situações em que a fala do utilizador é interpretada e consegue ser convertida num comando condição-ação.

Um exemplo comum que utilizamos com estes assistentes é a configuração de alarmes, isto é, quando pedimos ao assistente para definir um alarme para uma determinada hora, indiretamente o assistente vai criar uma regra em que a condição é o tempo e a ação é o tocar do alarme. Na figura 2.21 temos esquematizado a criação desta regra.

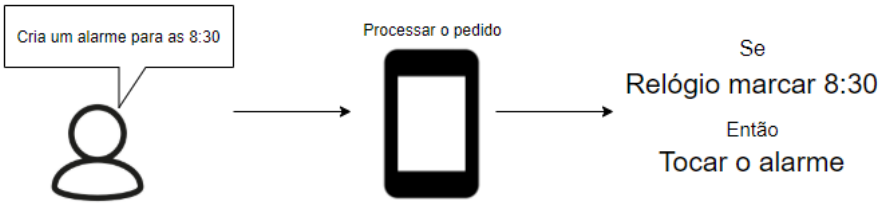


Figura 2.21: Exemplo das diversas partes de uma regra

Em [47] é dito que podemos pedir ao nosso assistente que toque música/rádio a uma hora específica do dia. Este pedido, como dito anteriormente pode ser convertido numa regra, onde neste caso, a ações é tocar música/rádio quando se verificar a condição da hora.

Em [48] foi desenvolvido um assistente pessoal utilizando um sistema de reconhecimento de voz que interpretava o que o utilizador dizia e conseguia extrair da voz palavras chave que eram posteriormente convertidas em comandos.

2.2.4 Texto

Além das interfaces apresentadas, também o texto pode ser utilizado para criar regras.

Numa atualização mais recente, o Home Assistant (falado no subcapítulo anterior), oferece aos utilizadores a possibilidade de criar automações através da escrita de uma frase. O assistente posteriormente vai processar a frase inserida e tentar convertê-la numa automação.

Na figura 2.22 podemos observar a interface que o Home Assistant fornece para converter uma frase numa automação.

Se o utilizador não quiser preencher todos os menus para a criação da automação neste assistente, pode simplesmente escrever o que pretende que aconteça no campo apresentado na figura 2.22 e posteriormente só necessita de indicar a que dispositivos se refere na frase.

2.2.5 BPMN - *Business Process Modeling Notation*

A notação BPMN, do inglês *Business Process Modeling Notation* é uma linguagem de modulação visual que foi criada para representar fluxos de forma a que qualquer pessoa pudesse visualmente interpretá-los e perceber os mesmos [49].

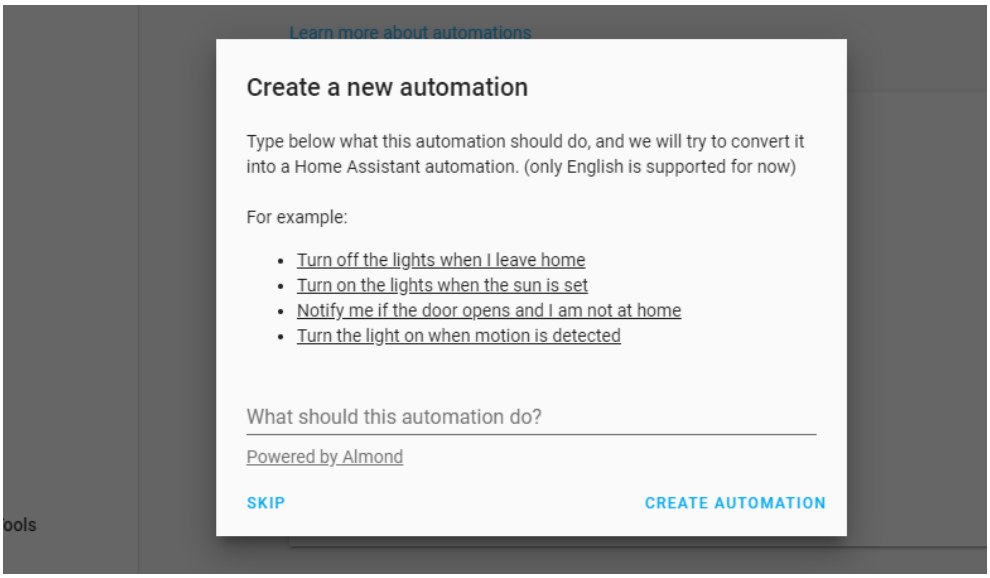


Figura 2.22: Interface do Home Assistant para conversão de uma frase em automação

Esta notação define um diagrama de processos de negócios que se baseia numa técnica de diagramas de fluxos. Estes diagramas são constituídos por elementos gráficos que utilizam formas diferentes para serem mais facilmente definidos, por exemplo, um rectângulo representa uma atividade e um losango uma decisão. Na figura 2.23 podemos visualizar um fluxo criado nesta notação.

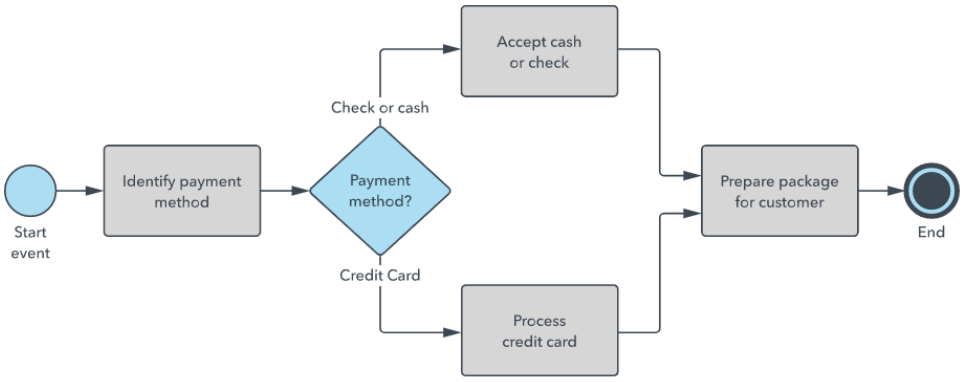


Figura 2.23: Fluxo utilizando a notação BPMN [50]

Podem ser identificados 4 tipos de elementos diferentes nos diagramas realizados nesta notação: [50]

1. Objectos de fluxo

   - **Atividades** - Representam uma tarefa que é realizada por um sistema ou pessoa;

- **Eventos** - Representam o ínicio, o fim ou modificam um processo;
- **Gateways** - Representam uma tomada de decisão baseada em condições ou eventos;

2. Objetos de conexão

   - **Sequência** - Mostra em que ordem as atividades estão a ser executadas. Simboliza-se com uma linha cheia com uma seta cheia;
   - **Mensagens** - Representam mensagens que variam entre dois processos. Simboliza-se com uma linha a tracejado com um círculo aberto e uma seta aberta;
   - **Associação** - Ligam os artefactos aos objetos de fluxo. Representa-se por uma linha a tracejado.

3. Swim lanes

   - **Piscinas** - Representam processos e os participantes num processo.
   - **Pista** - Uma piscina pode conter várias pistas, que representam várias áreas e responsabilidades no processo.

4. Artefactos - São utilizados para trazer mais detalhe para o diagrama.

2.2.6 CEP - *Complex Event Processing*

O CEP - Complex Event Processing [51] é um conjunto de técnicas para capturar e analisar conjuntos de dados e, à medida que chegam, tomar decisões consoante os dados que cheguem.

Na imagem 2.24 retirada de [51], podemos observar o diagrama de funcionamento deste método.

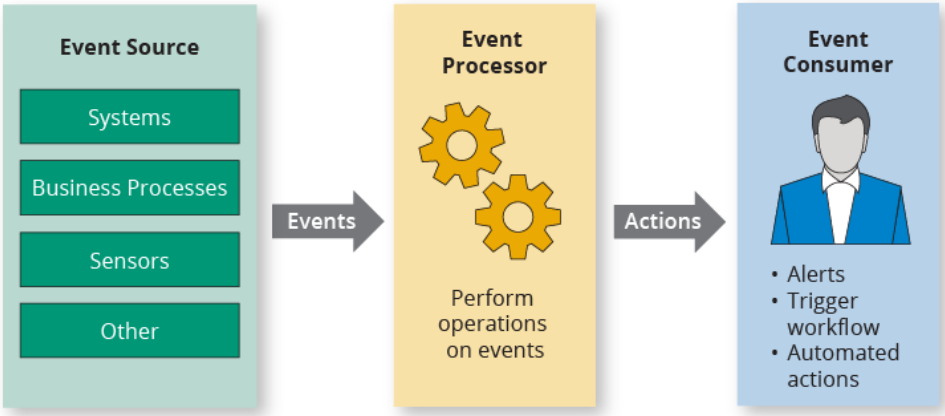


Figura 2.24: Diagrama do funcionamento do CEP [51]

O CEP é essencialmente associado à procura de padrões e dependências nos conjuntos de dados que lhe chegam.

De acordo com [52] o CEP pode ser utilizado nestas áreas:

- **Monitorização de Atividades de Negócio** - Ao analisar processos de negócios o CEP pode ser utilizado para detetar falhas ou oportunidades;
- **Monitorização de Dados de Sensores** - Analisando os dados provenientes de sensores, pode fazer desplotar ações de acordo com os valores transmitidos por esses mesmos sensores.
- **Monitorização dos Mercados** - Pode ser utlizado o CEP para monitorizar as subidas/descidas de preços de ações para saber qual é o melhor momento de comprar/vender.

2.3 Editores no mercado

2.3.1 Node-Red

O Node-Red[46] é provavelmente o motor de regras mais utilizado atualmente, oferecendo uma integração com o Home-Assistant. Para fazer a integração com o Home-Assistant requer que o utilizador tenha o minímo de conhecimento de administração de sistemas, pois a sua integração não é intuitiva, tendo de se percorrer as definições do assistente para encontrar e instalar o Add-On do Node-Red. O seu desenvolvimento é acente em Node.js, o que permite a adição de módulos novos, oferecendo uma vasta coleção de novas funcionalidades.

O Node-Red é utilizado pelos utilizadores que têm algumas dificuldades com a escrita dos ficheiros no formato YAML então recorrem ao Node-Red e configuram as suas automações através desta plataforma.

Os fluxos criados no Node-Red podem ser exportados para o formato JSON o que possibilita a partilha de fluxos entre a comunidade [53].

Na figura 2.25, temos uma pequena apresentação da interface do NodeRed. No lado esquerdo encontram-se a palete de componentes onde, como foi dito, se pode fazer a adição de componentes e, do lado direito, uma tabela de informações sobre o nó selecionado.

Um dos principais problemas do Node-Red está relacionado com a facilidade de interação com o utilizador. Apesar de ter um vasto leque de componentes disponíveis, a sua configuração é bastante complexa para um utilizador que apenas pretende criar umas simples automações para ter na sua habitação.

2.3.2 Particle

O motor de regras do Particle é um construtor de workflows *if-then* especialmente desenhado para dados de IOT. Ele permite que cada pessoa possa criar facilmente lógicas condicionais utilizando uma interface visual [55].

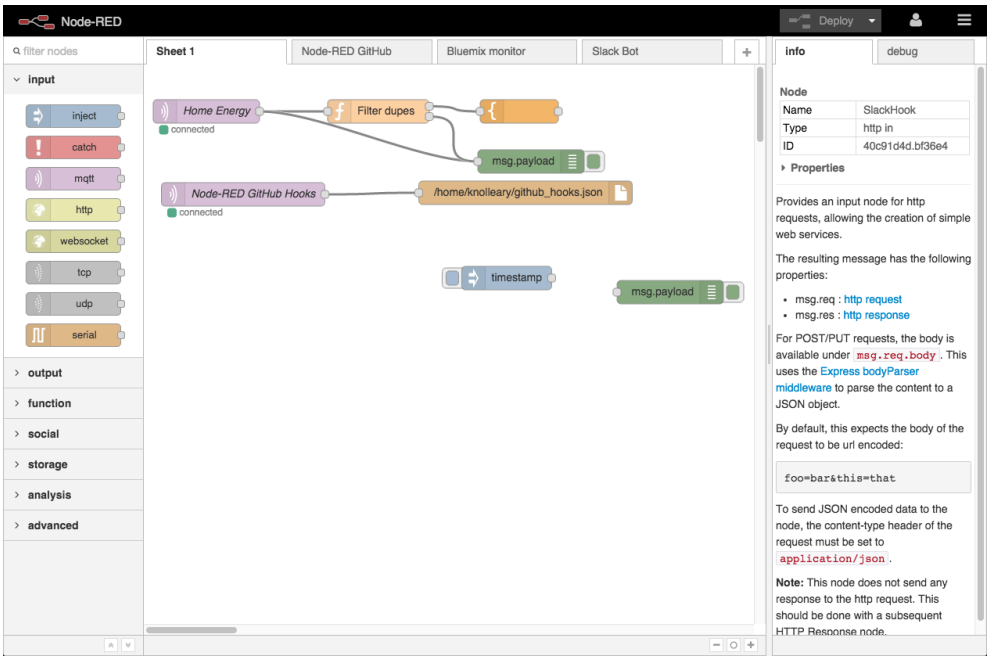


Figura 2.25: Interface da plataforma NodeRed [54]

Este motor de regras é construído em cima do Node-Red logo, oferece as mesmas funcionalidades. Para além de apresentar as mesmas funcionalidades, faculta uma interface um pouco mais cuidada mas, mesmo assim, muito díficil de utilizar para uma pessoa comum que não está completamente inteirada do sistema.

Como é contruído e baseado no Node-Red, também ele pode ser utilizado por utilizadores que têm dificuldades na construção de ficheiros YAML.

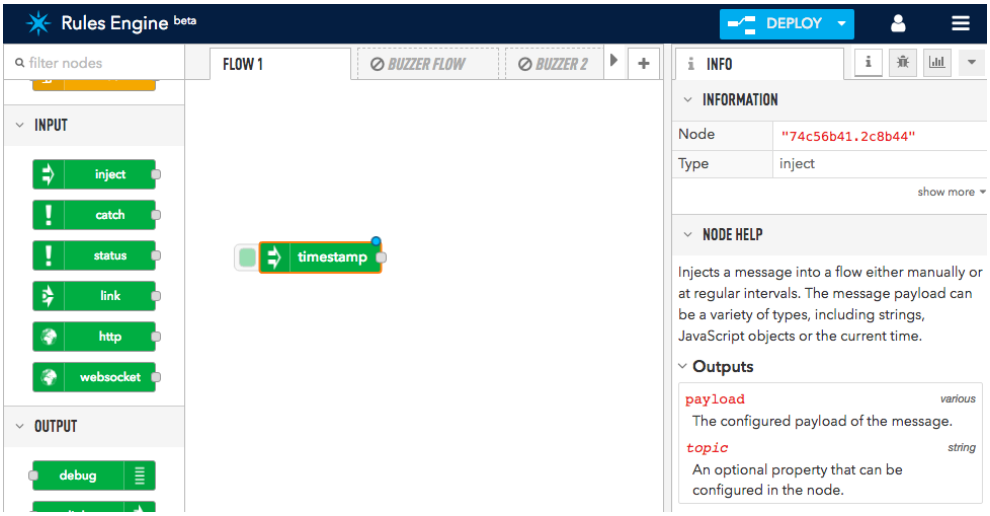


Figura 2.26: Interface da plataforma Particle [55]

Tal como o Node-Red oferece também ele uma palete de componentes, que o utilizador pode utilizar como bem entender, podendo também acrescentar novos componentes ao motor de regras.

Na figura 2.26 temos uma visualização do Particle. Como se pode observar é bastante parecido ao Node-Red mas, aparentemente, como uma interface um bocado mais *user-friendly*. Os problemas adjacentes ao Particle são os mesmos, problemas estes que passam pela dificuldade que o utilizador tem em trabalhar com o sistema.

2.3.3 ThingsBoard

O ThingsBoard é uma plataforma de recolha, processamento e visualização de dados e também de gestão de dispositivos, o que o coloca no mesmo nível do Home-Assistant e de outros assistentes [56][57].

Esta plataforma oferece um editor próprio que é inteiramente direcionado para processamento de eventos complexos. Com este editor é possível filtrar, enriquecer e transformar mensagens provenientes de vários dispositivos de IOT.

Nesta plataforma é também possível criar nós com diversas funções específicas, mas para isso é necessário que o utilizador realize a configuração desses mesmos nós o que, para um utilizador comum pode ser uma tarefa complicada de realizar.

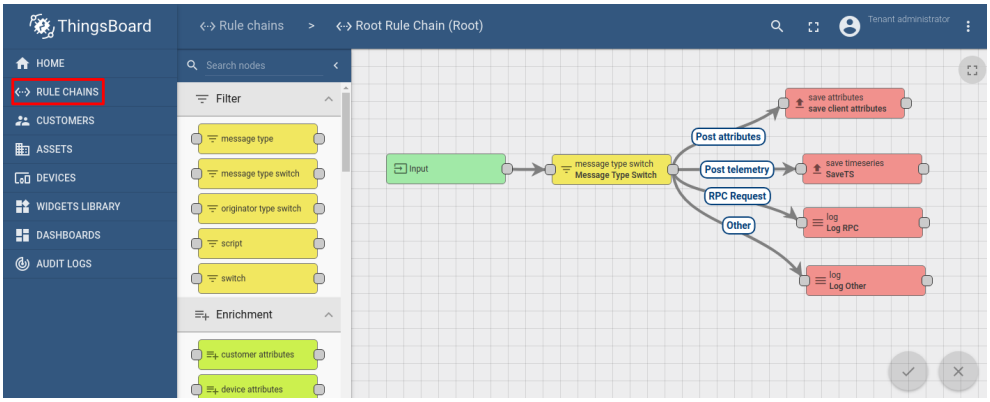


Figura 2.27: Interface do editor do ThingsBoard [56]

Na figura 2.27 podemos visualizar o editor que está incorportado nesta plataforma. Conseguimos observar e constatar que o seu aspeto é bastante parecido aos editores anteriormente apresentados.

Foram apresentadas diversas alternativas às quais o utilizador pode recorrer para criar as suas automações de forma mais simples. Contudo, todas as alternativas têm as suas limitações que, na generalidade, passa pela dificuldade que um utilizador comum tem para efetuar as configuações necessárias.

	Node-Red	Particle	ThingsBoard
Facilidade na configuração dos componentes	✗	✗	✗
Progamação com fluxos	✓	✓	✓
Acesso simples através da Web sem necessária instalação	✗	✗	✗
Adição de componentes ao editor de forma rápida	✓	✓	✓
Configuração de forma rápida com diversos assistentes	✓	✓	✗

Tabela 2.1: Tabela comparativa entre os vários editores

2.3.4 Comparação

Entre os editores apresentados e estudando todas as suas características principais, foi possível a criação de uma tabela que compara os vários editores. Na tabela 2.1 podemos observar essas comparações.

2.4 Controlo de erros

Tendo analisado o estado de arte no que diz respeito a interfaces de criação de automações e editores existentes atualmente no mercado com o objetivo de facilitar a criação dessas mesmas automações, foi realizado um estudo como poderia ser feito um controlo de erros. Para esse controlo de erros, duas alternativas distintas podem ser utilizadas:

- **Plataformas de Controlo de Erros**: Estas plataformas não necessitam da interação do utilizador. São configuradas no código fonte das aplicações no qual pretendemos o controlo de erros e sempre que um erro surge a plataforma é alertada e regista o erro que foi reportado. Desta forma o utilizador apenas tem de utilizar normalmente a plataforma pois, caso um erro aconteça, a plataforma é alertada.

- **Formulários**: As plataformas podem oferecer ao utilizador um formulário onde estes podem indicar possíveis erros que encontraram durante a utilização. As respostas a estes formulários são analisadas e erros que possam originar esses problemas são corrigidos.

2.4.1 Plataformas de Controlo de Erros

Tentar obter/controlar todo o tipo de erros que uma plataforma sofre pode-se tornar muito complicado devido ao facto de um erro poder acontecer raramente ou então por não estarmos

sempre a observar os logs provenientes da mesma [58].
Neste subcapítulo serão apresentadas plataformas que, como referido acima, conseguem realizar um controlo de erros sem necessária intervenção por parte do utilizador. Nestas plataformas temos o exemplo do Sentry, do Bugsnag e do Rollbar.

Sentry

O Sentry [59] é uma das plataformas existentes no mercado para fazer monitorização de erros.

A facilidade de configuração desta plataforma é grande, apenas sendo necessário que o programador crie um projeto para a aplicação da qual pretende monitorização, instancie o Sentry no código fonte da sua aplicação e depois a API do Sentry deteta os erros e envia-os para página do seu projeto na plataforma. No caso de uma aplicação web, o Sentry pode ser instanciado como sugere a figura 2.28.

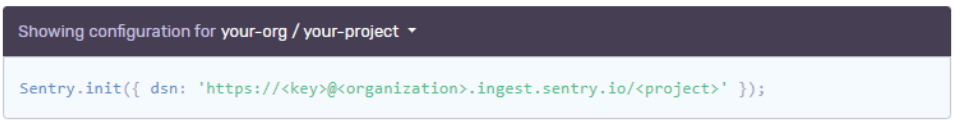


Figura 2.28: Instalação do Sentry para uma aplicação web [59]

Os criadores do Sentry oferecem uma versão com um plano gratuito e oferecem na sua página toda a documentação necessária para efetuar a monitorização do seu código. Se pretender instalar o Sentry num servidor pessoal, também existe documentação para essa alternativa.

No que diz respeto a plataformas suportadas, o Sentry suporta todo o tipo de linguagens de progamação e *frameworks* que vão desde C, JavaScript, Python, Java, Ruby a Django, Angular, Flask, React entre muitas outras alternativas [60].

Sempre que algum erro é captado pelo Sentry é agrupado com outros erros de eventos similares, ou seja, o utilizador não necessita de agrupar os erros por tipo de evento despoletador de erros, uma vez que o Sentry já faz esse trabalho. Na figura 2.29 podemos observar a interface onde o Sentry apresenta os erros que capta. Note-se que em cada erro/problema é apresentado o número de eventos que já ocorreram que originaram um erro daquele tipo.

A cada erro são associados dados que podem ser relevantes na correção dos mesmos. Esses dados contêm várias informações dentro das quais se incluem os sistemas operativos que o utilizador estava a utilizar, o software utilizado para aceder entre outros.

Para além de captar os erros, o Sentry tambem oferece uma interface para os utilizadores darem o seu feedback podendo assim juntar dados provenientes destas duas fontes: os dados captados automaticamente e os feedbacks dos utilizadores.

Se a equipa de desenvolvimento estiver dividida em subequipas é possível atribuir um erro

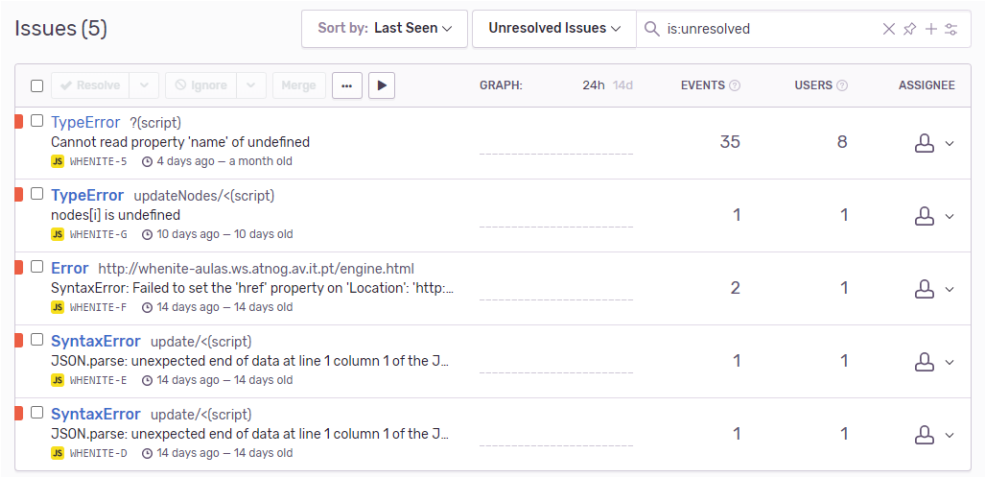


Figura 2.29: Interface de apresentação de erros do Sentry

a uma certa equipa, isto é, se o erro pertencer ao componente X da aplicação, esse erro é atribuído à equipa responsável por esse componente. Este sistema facilita na resolução de problemas.

O Sentry oferece também inúmeras integrações que vão desde o Slack, GitHub, Trello, Jira, entre outros.

Bugsnag

O Bugsnag [61] é outra plataforma que tem o mesmo propósito do Sentry, a monitorização de erros em código.

Tal como a plataforma anteriormente mencionada, também o Bugsnag tem uma elevada facilidade na sua configuração e instalação como apresentado na figura 2.30.

```
Bugsnag.start({
  apiKey: 'YOUR_API_KEY',
  otherOptions: value
})
```

Figura 2.30: Instalação do Bugsnag para uma aplicação web [61]

Oferece controlo de erros para um enorme número de linguagens de programação em que se inclui JavaScript, Android, Java, Go, PHP, Python, Ruby, entre outras.

Permite a atribuição de papéis a cada uma das pessoas que se encontram a trabalhar no projeto, sendo que podem ser atribuídos o papel de administrador e o papel de colaborador. O administrator tem todas as permissões nos projetos em que está inserido que vão desde criar ou apagar um projeto, atribuir papéis, gerir o acesso dos colaboradores, configurar integrações ou fazer uma gestão dos erros. Por outro lado os colaboradores têm permissões um pouco mais

restringidas podendo fazer uma gestão dos erros que possam surgir e configurar integrações aos projetos onde se encontram.

Sempre que um erro é detetado pelo Bugsnag, um email é enviado a alertar. Podemos obter mais informações acerca do ambiente em que o erro foi captado, o que pode permitir aos programadores detear mais rapidamente uma possível origem do erro ou até mesmo informações sobre o erro em si. É possível atribuir um determinado erro a um colaborador em específico.

O Bugsnag oferece uma enorme quantidade de possíveis integrações que podemos acrescentar ao nosso projeto que vão desde integrações para notificação de equipas (Slack, por exemplo) a integrações para *tracking* de problemas (Jira, por exemplo).

Podemos nesta plataforma agrupar os erros de diversas maneiras, desde a classe do erro até à linha que originou um erro específico.

Na figura 2.31 retirada do blog do Bugsnag [62] temos uma imagem da interface do Bugsnag, onde são apresentados os erros que a plataforma conseguiu captar.

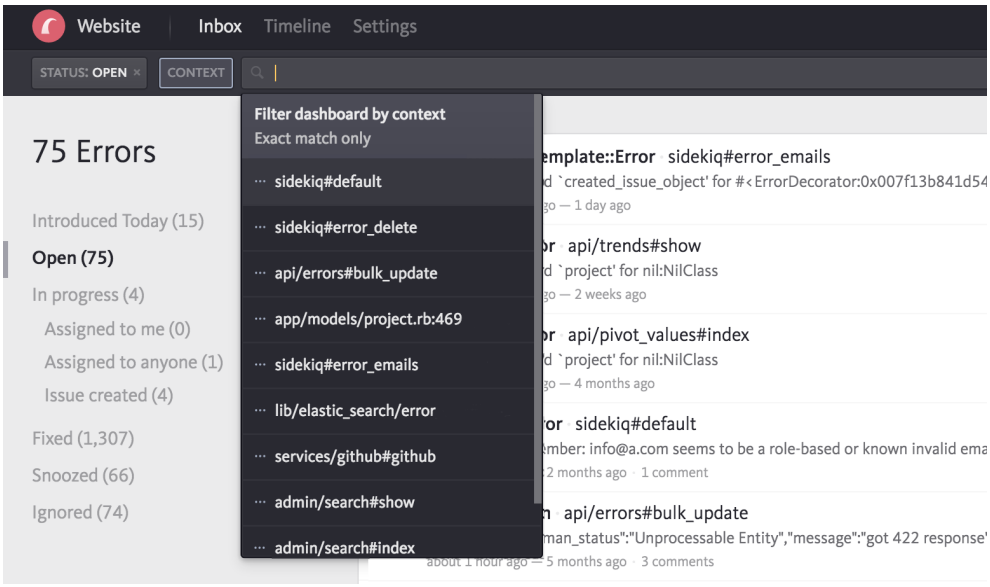


Figura 2.31: Interface de apresentação de erros do Bugsnag [62]

Rollbar

O Rollbar [63] mostra-se como uma alternativa que também desempenha um papel na monitorização e deteção de erros, tal como as plataformas mencionadas anteriormente.

A instalação do Rollbar é bastante simples. Apenas é necessário a criação de um projeto consoante a linguagem em que é programada a nossa aplicação e posteriormente apenas temos de instanciar o Rollbar na nossa aplicação como está presente na figura 2.32. Após criar a instância, ele é capaz de captar erros que surjam e listá-los na página da plataforma.

Figura 2.32: Instalação do Rollbar para uma aplicação web [63]

São suportadas várias linguagens de programação sendo algumas delas Java, JavaScript (seja ele no Browser ou em Node.js), Python, PHP, Ruby, entre outras.

Os erros podem ser encontrados em três estados diferentes: ativos, resolvidos e silenciados.

Os erros ativos são aqueles que são detetados e sendo assim aparecem na dashboard da plataforma, sendo que estes erros notificam os colaboradores do projeto aquando da sua primeira ocorrência e quando algumas condições são registadas, que podem ser configuradas. Os erros resolvidos, como o nome indica, é o estado em que o erro se encontra à partida corrigido e já não sem verifiquem mais ocorrências daquele erro. Estes erros já não são apresentados na dashboard.

Por fim temos os erros que se encontram no estado silenciado. Este estado é utilizado se o tipo de erros associado não está sob inventigação mas os erros continuam a poder se processados e pesquisados. A qualquer momento, um erro no estado silenciado pode ser reaberto passando ao estado ativo.

No que diz respeito a integrações, é possível integrar no Rollbar inúmeras aplicações como é o caso do Slack no caso de notificações ou como os casos do Jira e do Trello para fazer *tracking* de problemas.

Na figura 2.33 retirada do website da plataforma temos uma imagem da sua interface de apresentação de erros

Comparação

Como apresentado nos subcapítulos anteriores, existem inúmeras plataformas que permitem a captação, registo e controlo de erros de diversas linguagens de programação para todo o tipo de aplicações.

Na figura 2.34 temos apresentado o número de downloads de cada uma aplicações entre maio do ano 2019 e maio do ano de 2020. Podemos constatar que o Sentry é a plataforma mais utilizada para monitorização de erros seguido do Rollbar e por fim o Bugsnag.

A curva do Sentry apresenta uma subida constante ao longo do tempo tirando o período próximo a janeiro do ano corrente, onde houve uma queda do número de downloads mas, mesmo assim, mantendo-se acima das outras plataformas.

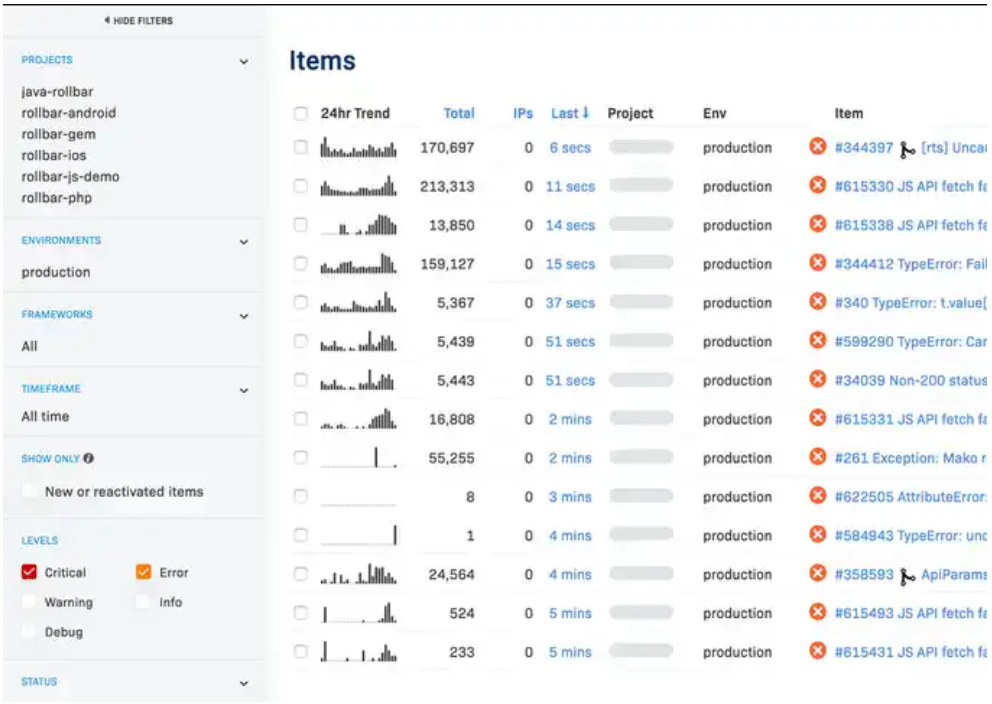


Figura 2.33: Interface de apresentação de erros do Rollbar [63]

As curvas do Rollbar e do Bugsnag mantiveram-se contantes ao longo do tempo apesar de, à semelhança do Sentry, ter havido uma queda por volta de mês de janeiro de 2020.

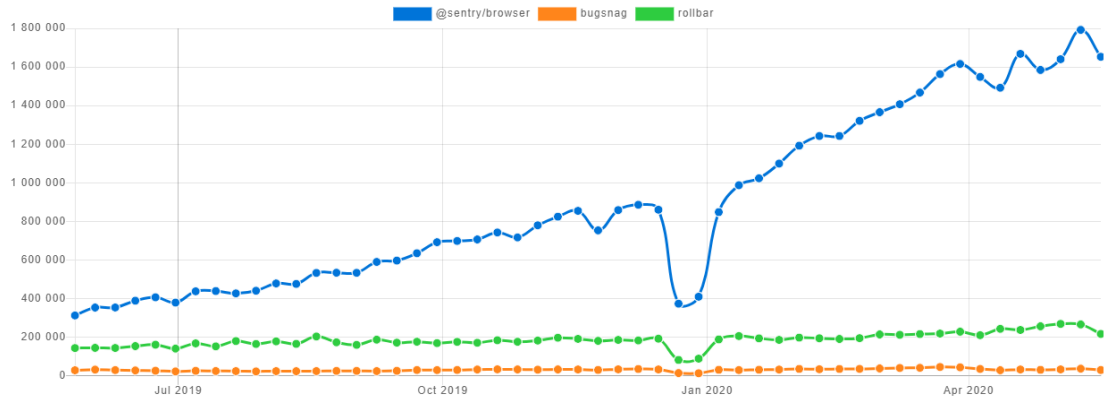


Figura 2.34: Downloads de cada uma das plataformas descritas entre maio de 2019 e maio de 2020 [64]

Capítulo 3

Arquitetura

Após ser apresentado o estado da arte no que diz respeito a editores aos quais os utilizadores recorrem para facilitar a criação das suas automações, foi possível traçar um plano para a criação de uma solução capaz de colmatar falhas nos editores atualmente existentes no mercado.

Durante este capítulo serão apresentados os requisitos que a plataforma deverá cumprir e posteriormente apresentada uma arquitetura para a mesma.

3.1 Requisitos

Foram traçados alguns requisitos que a plataforma deveria ter de modo a poder concorrer com as atuais plataformas.

Na tabela 3.1 podemos ver alguns dos requisitos que a plataforma deveria cumprir para realizar o seu principal objetivo, a simplificação na criação de automações no Home Assistant.

3.2 Casos de uso

Tendo em conta os requisitos apresentados foi possível analisar e retirar desses requisitos alguns casos de utilização que estarão associados à plataforma. Posto isto, de seguida serão apresentados alguns dos casos de uso da plataforma pretendida e uma explicação dos mesmos.

- **Conectar com Home Assistant**: O utilizador tem de ser capaz de conectar com o seu assistente de modo a que a plataforma consiga identificar os dispositivos que se encontram ligados ao mesmo

- **Compilar o diagrama**: O utilizador, após criar a automação pretendida no editor, tem de conseguir compilar o mesmo de forma a extrair o código *yaml* correspondente.

Requisito	Explicação
Plataforma Web	A plataforma teria de ser web, para chegar ao maior número possível de utilizadores e não necessitar de configurações.
Ligação com o Assistente	A plataforma tem de permitir uma conexão ao assistente.
Programação com fluxos	Tem de oferecer uma programação com fluxos que permita desenhar um diagrama representativo da automação.
Geração de código	Possibilitar a geração de código correspondente ao diagrama desenhado
Guardar Automações	De modo a permitir ao utilizador guardar a sua automação para poder continuar mais tarde a sua edição.
Download e upload de automações	O utilizador tem de ser capaz de fazer o download diagramas ou carregar diagramas se tiver algum no seu computador.

Tabela 3.1: Tabela explicativa dos requisitos que a plataforma deveria cumprir

- **Guardar Automação**: Caso o utilizador queira continuar a edição da sua automação mais tarde ele pode guardar a mesma.
- **Ver automações guardadas**: Caso tenha guardado alguma automação, o utilizador tem de conseguir visualizar todas as automações que foram previamente guardadas
- **Fazer download do diagrama**: O utilizador pode fazer download de diagramas associados a automações para o seu computador.
- **Carregar diagrama**: Se tiver no seu computador algum diagrama representativo de uma automação, pode carregar esse diagrama para o editor e realizar as alterações que necessite.

Identificados alguns dos casos de uso, na figura 3.1 temos esquematizado o diagrama de casos de utilização da plataforma para o utilizador final.

3.3 Arquitetura

A arquitetura pretendida para o sistema foi pensada sendo o mais modular possível, isto é, que permitisse facilmente alterações sem serem necessárias muitas alterações ao código da plataforma. Com uma arquitetura modular, podem-se acrescentar facilmente novas funções à plataforma. Nesta arquitetura da plataforma podemos constatar 4 módulos diferentes diferentes, como é possível visualizar no diagrama de componentes apresentado na figura 3.2:

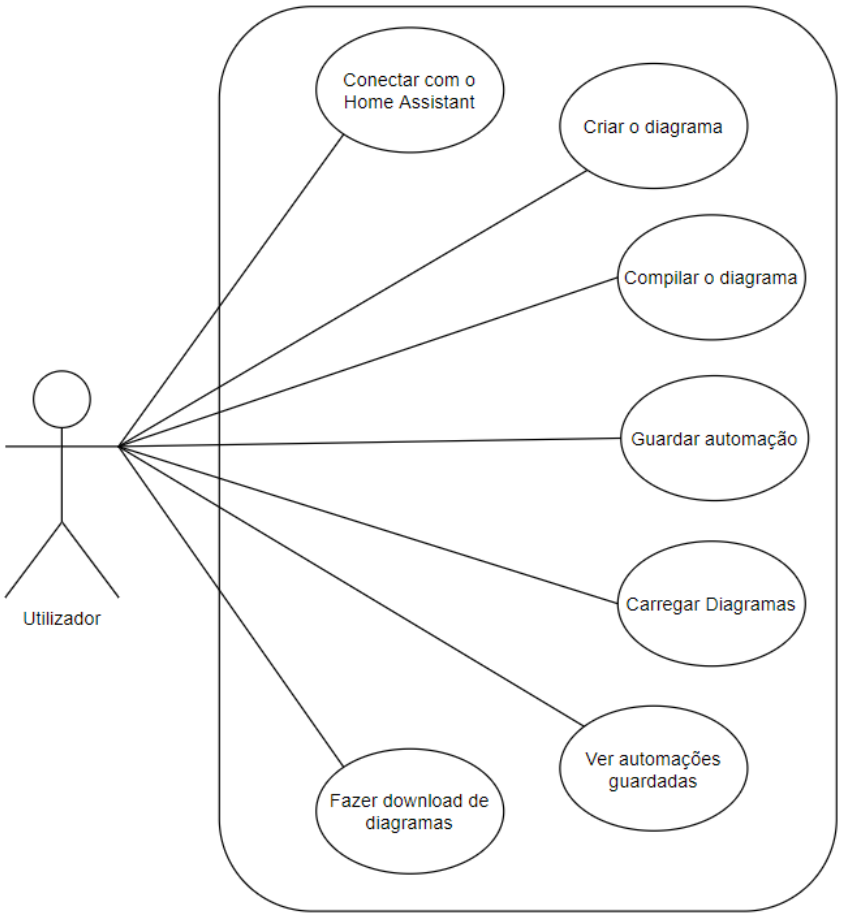


Figura 3.1: Diagrama de casos de uso da plataforma

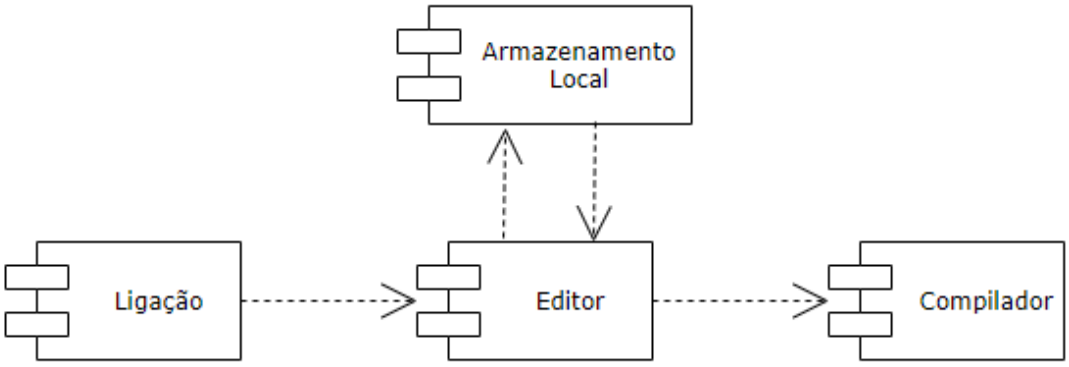


Figura 3.2: Diagrama de componentes da plataforma

- **Ligação**: Este módulo é responsável por estabelecer a conexão entre a plataforma com o seu assistente e recolher os dados relativos aos dispositivos conectados para enviar ao

editor;

- **Editor**: Módulo responsável pela criação do fluxo da automação. Adicionando os dispositivos que pretendemos para a automação ao editor e estabelecendo conexões entre eles de modo a criar um fluxo;
- **Armazenamento Local**: Este módulo tem como objetivo servir de armazenamento para as automações que são guardadas por parte do editor;
- **Compilador**: O compilador gera o ficheiro correspondente da automação que foi criada no editor, possibilitando ao utilizador o *copy-paste* do código correspondente para poder acrescentar nos ficheiros de configuração do seu assistente.

3.4 Esboço da plataforma

Tendo em conta a arquitetura pensada e toda a pesquisa apresentada no capítulo 2, foi traçado um esboço da plataforma pretendida de maneira a cumprir com todas as necessidades da melhor maneira possível. Seria necessário na plataforma uma zona onde a adição de dispositivos ao editor seria de fácil acesso e seria necessário uma área onde o utilizador pudesse guardar ou compilar os seus diagramas.

Na figura 3.3 podemos observar o esboço pensado para a plataforma.

3.5 Modularidade da plataforma

Apesar da plataforma desenvolvida ter sido preferencialmente desenhada para facilitar a criação de automações para o Home Assistant, a sua arquitetura permite que possa ser usada para outros assistentes como por exemplo o openHab, apresentado no capítulo 2. Apenas são necessárias alterações ao nível do login através da utilização das APIs do assistente pretendido, e alterações ao nível do compilador pois, como observámos no capítulo 2, o ficheiro que contém as automações varia entre cada assistente.

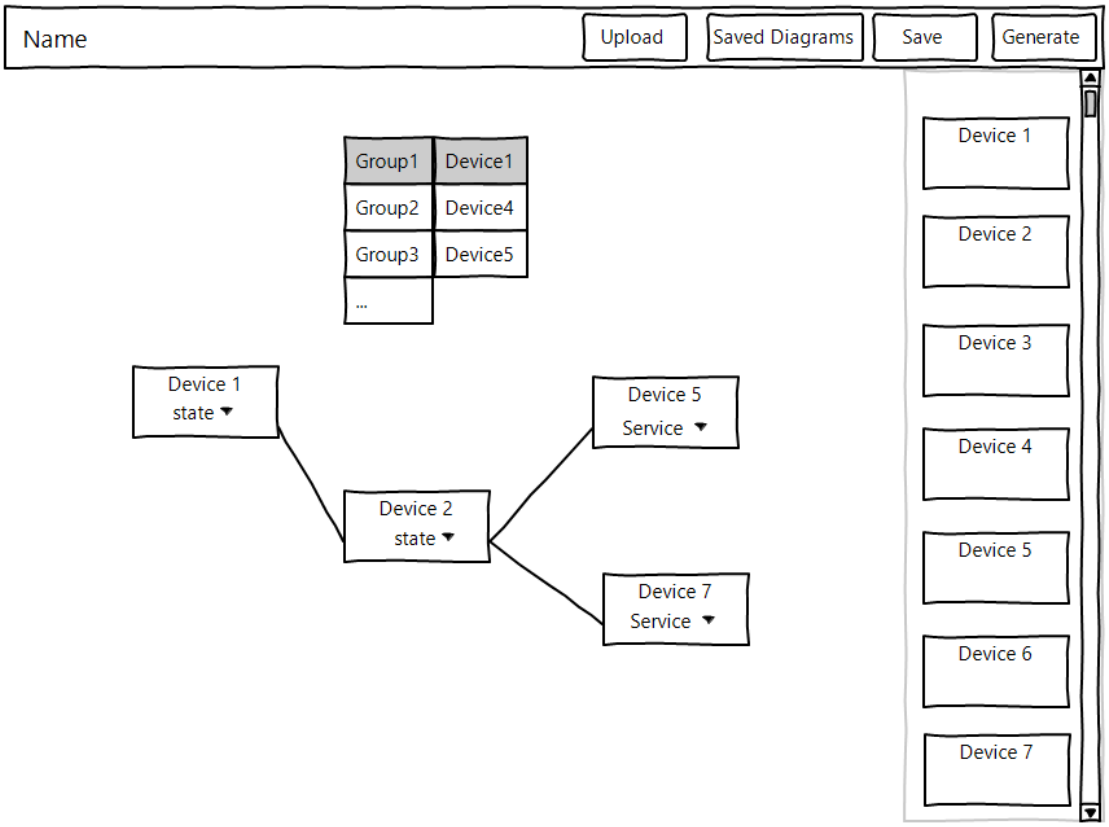


Figura 3.3: Esboço da plataforma pretendida

Capítulo 4

Implementação

Tendo em conta a arquitetura e os requisitos apresentados no capítulo anterior, foi possível começar a implementar esta nova plataforma.

Ao longo deste capítulo é introduzida a plataforma identificando e explicando os componentes que a constituem e como foram criados cada um deles ao nível do código.

4.1 Whenite

O Whenite foi o editor criado para colmatar todos os problemas que as plataformas existentes atualmente apresentam. Esta nova plataforma foi desenhada para o Home Assistant, uma vez que este assistente é um dos mais utilizados para automação residencial e também por ter alargadas e reconhecidas comunidades mundiais. Após um estudo sobre as atuais plataformas existentes no mercado foi possível apontar pequenos problemas que estas apresentavam e tentar contornar esses problemas com uma abordagem um pouco diferente. A ideia base era a criação de uma plataforma Web, capaz de ser acedida por todos os utilizadores, sem qualquer intalação ou incorporação no seu Home Assistant através de *add-on* pois, para uma pessoa com reduzido conhecimento de administração de sistemas, seria um pouco complexo de realizar.

4.2 Ligação

Autenticação com Home Assistant

Quando o utilizador entra na página do editor da plataforma é-lhe pedido o endereço IP onde o seu assistente se encontra. Se estiver a aceder à plataforma na rede da sua habitação pode colocar o IP da rede da habitação. Por outro lado, caso esteja a tentar aceder à sua instâncida do Home Assistant pela internet (através de um endereço do DuckDNS [65] ou similar) necessita de introduzir o url definido para a sua instância.

Após a inserção do IP, o utilizador é reencaminhado para a página de login do seu Home Assistant, como podemos observar na figura 4.1. Na figura pode-se ser que estamos a facultar o acesso da plataforma do Whenite à nossa instância do Home Assistant.

Depois de inseridas corretamente as credenciais de acesso, automaticamente é redirecionado para o editor, editor este que é automaticamente populado pelos dispositivos que se encontram ligados ao assistente.

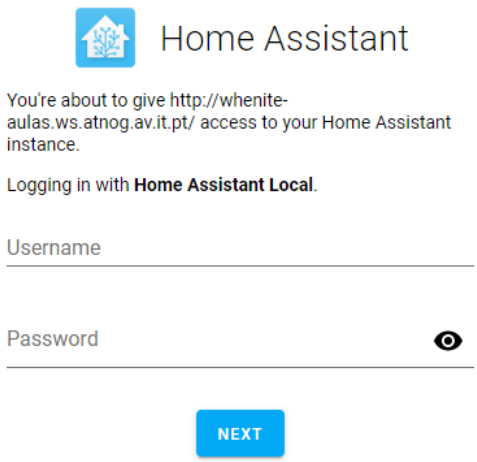


Figura 4.1: Página de autenticação do Home Assistant

Utilização da API

Após o login efetuado com sucesso, é utilizada a *Websocket API* [66] do Home Assistant para ser possível obter as entidades que se encontram ligadas ao assistente bem como os serviços que se encontram associados a cada tipo de entidade.

Para a obtenção das entidades e dos serviços foram testadas duas abordagens distintas, acabando por ser optada a utilização da segunda abordagem.

Inicialmente foi utilizada a API Rest [67] mas foram encontrados problemas nesta API quer pelo conteúdo que a mesma era capaz de facultar quer pela difícil tarefa que teria de ser pedida ao utilizador para realizar para que a API funcionasse corretamente. Esta tarefa passava pela geração de um *Bearer Token* que teria de ser usado no *header* do pedido feito para a API. O outro problema associado devia-se ao facto desta API não disponibilizar as entidades que estavam ligadas ao assistente, somente disponibilizava os serviços associados às mesmas.

Como um dos objetivos da plataforma era simplificar a interação do utilizador, pedir-lhe para gerar um *bearer toker* seria algo complicado, principalmente para pessoas como poucos conhecimentos de administração de sistemas.

Posto esta primeira abordagem, foi utilizada a *Websocket API* do Home Assistant. Esta API, além de utilizar a autenticação por credenciais do assistente, era capaz de disponibilizar quer as entidades quer os serviços associados.

Respostas da API

Quando utilizada a API, as respostas vêm em formato JSON como se pode observar na figura 4.2.


```
"switch":{
  "turn_off":{
    "description":"Turn a switch off.",
    "fields":{
      "entity_id":{
        "description":"Name(s) of entities to turn off.",
        "example":"switch.living_room"
      }
    }
  },
  "turn_on":{
    "description":"Turn a switch on.",
    "fields":{
      "entity_id":{
        "description":"Name(s) of entities to turn on",
        "example":"switch.living_room"
      }
    }
  },
  "toggle":{
    "description":"Toggles a switch state.",
    "fields":{
      "entity_id":{
        "description":"Name(s) of entities to toggle.",
        "example":"switch.living_room"
      }
    }
  }
},
```

```
"binary_sensor.office_window":{
  "entity_id":"binary_sensor.office_window",
  "state":"off",
  "attributes":{
    "friendly_name":"Office Window",
    "device_class":"opening"
  },
  "last_changed":"2020-05-19T10:45:30.589149+00:00",
  "last_updated":"2020-05-19T10:45:30.589149+00:00",
  "context":{
    "id":"3d043a3c73314e46b26cebf300776106",
    "parent_id":null,
    "user_id":null
  }
},
"switch.sonoff_office":{
  "entity_id":"switch.sonoff_office",
  "state":"on",
  "attributes":{
    "friendly_name":"SonOff Office ",
    "assumed_state":true
  },
  "last_changed":"2020-05-19T10:45:30.591611+00:00",
  "last_updated":"2020-05-19T10:45:30.591611+00:00",
  "context":{
    "id":"3ef47617c19c4f9dad3fb8963213295d",
    "parent_id":null,
    "user_id":null
  }
```


Figura 4.2: Respostas da API do Home Assistant. À esquerda os serviços e à direita as entidades

À esquerda temos a resposta para o pedido das entidades ao Home Assistant, isto é, os dispositivos que se encontram ligados. À direita, temos os serviços que cada domínio pode apresentar. Para o caso de um interruptor por exemplo, o seu domínio é *switch* e apresentam 3 tipos de serviços: **turn off**, **turn on** e **toggle**.

4.3 Editor

Após a fase do componente da ligação, o editor já tem disponíveis para si todos os dispositivos conectados ao assistente para o utilizador poder utilizar na criação de um fluxo que mais tarde será convertido em código.

De seguida, é apresentado com mais detalhe o componente do editor, como apresentado no

capítulo anterior. Neste componente é utilizada uma framework para programação em fluxos às quais foram adicionados plugins. Foi com o recurso a estes plugins que foi possível oferecer no editor um vasto leque de possibilidades a nível de inserção de dispositivos no editor ou até mesmo a possibilidade de se conectarem os diversos dispositivos para criar o fluxo pretendido.

Na figura 4.3 está esquematizado o bloco do editor e de seguida será feita uma explicação mais aprofundada de cada um dos componentes e o modo como foi implementado.

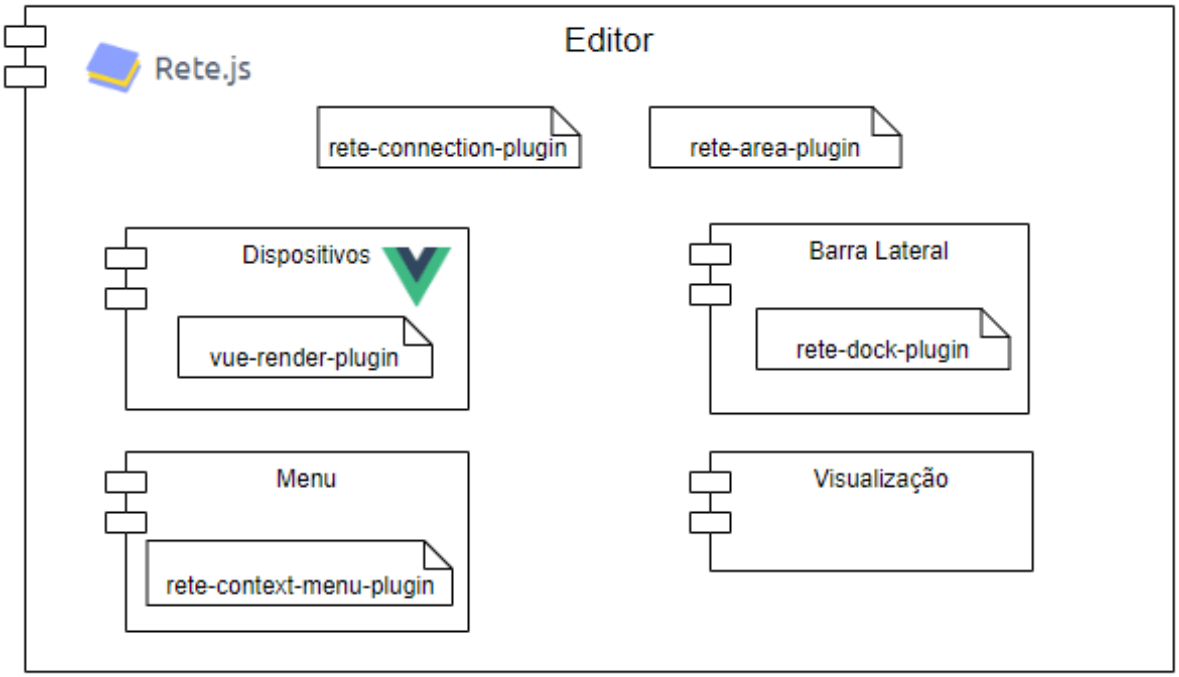


Figura 4.3: Diagrama de Componentes do Editor

4.3.1 Frameworks

Ao longo deste subcapítulo numa primeira fase serão apresentadas algumas das frameworks que foram testadas quando foi realizado o levantamento dos requisitos. Posteriormente será indicada qual a framework escolhida.

Frameworks testadas

Inicialmente a procura foi baseada em frameworks que possibilitassem a programação baseada em fluxos e surgiram algumas alternativas nomeadamente o Total.js [68], o Blockly [37] e o Rete.js [69].

A framework Total.js é uma framework para Node.js orientada ao tema da *Internet of Things* que possibilita a criação de fluxos entre diversos componentes. Um dos problemas

desta framework prende-se no facto de, como um dos requisitos passava por ser uma plataforma simples na web, a utilização de Node.js para a criação do editor não seria uma boa ideia.

Como dito no capítulo 2, o Blockly permite a criação de código virtual, o que ajuda os utilizadores com menos conhecimentos ao nível da programação a criar código. Esta alternativa oferece uma interface que não é muito apelativa e organizada, o que pode confundir os utilizadores se existirem muitos elementos no editor.

Framework utilizada

A framework utilizada para a elaboração do editor foi o Rete.js. Uma vez que esta é uma framework baseada em Javascript muito flexível, podemos criar os componentes da forma que pretendermos sejam com formas ou cores diferentes. Além de flexível é uma plataforma modular, o que permite a adição de novas funcionalidades através de *plugins*.

Ao longo da criação do editor foram utilizados diversos plugins que ajudaram a facultar ao utilizador componentes do editor para uma mais simples usabilidade da plataforma.

4.3.2 Criação de Componentes

No que diz respeito à criação dos componentes, foram criados 3 tipos de componentes sendo eles para o registo horário, para sensores e outro tipo para os restantes dispositivos.

Os componentes foram desenhados com recurso a Vue.js [70], uma *framework* javascript. Esta *framework* é adicionada ao Rete.js por incorporação de um plugin denominado de *vue-render-plugin*. Quando foi construído o componente através do Rete.js, é indicado no seu construtor o *template* do Vue.js que é pretendido para os aspeto daquele componente. Na figura 4.4 temos o código responsável pela criação dos componentes.

```
var CustomNode = {
  template: `<div class="node" :class="[node.name ] | kebab">
  <div class="title" style="text-align:center" >{{node.name}}</div>
  <!-- Controls--><center>
  <div class="control" v-for="control in controls()" v-control="control"></div>
  </center>
  <!-- Outputs-->
  <div style="float:right" class="output" v-for="output in outputs()" :key="output.key">
    <div class="output-title">{{output.name}}</div>
    <Socket v-socket:output="output" type="output" :socket="output.socket"></Socket>
  </div>
  <!-- Inputs-->
  <div class="input" v-for="input in inputs()" :key="input.key">
    <Socket v-socket:input="input" type="input" :socket="input.socket"></Socket>
    <div class="input-title" v-show="!input.showControl()">{{input.name}}</div>
    <div class="input-control" v-show="input.showControl()" v-control="input.control"></div>
  </div>
</div>`,
  mixins: [VueRenderPlugin.mixin],
  components: {
    Socket: VueRenderPlugin.Socket
  }
}
```

```
class Component extends Rete.Component {

    constructor(name){
        super(name);
        this.data.component = CustomNode;
    }

    builder(node) {
        var out = new Rete.Output('out', "", socket);
        var in1 = new Rete.Input('inp', "", socket,true);
        return node.addOutput(out).addInput(in1);
    }

    worker(node, inputs, outputs) {
        outputs['out'] = node.data.num;
    }

}
```

Figura 4.4: Construção de um componente com recurso a Vue.js

Na criação do componente, é adicionada uma saída e uma entrada (no Rete.js são denominados de sockets), que são responsáveis por permitir a conexão entre componentes de modo

a criar um fluxo.

À medida que a automação for construída no editor, quando um componente é identificado como um *trigger*, uma condição ou uma ação, é adicionado ao componente um *control* diferente consoante o tipo de componente que é.

Após criados os componentes de acordo com os dispositivos conectados ao assistente é necessário oferecer ao utilizador a possibilidade de adicionar os componentes desejados ao editor para posteriormente começar a criar a automação desejada. Para a adição dos componentes ao editor, são oferecidas pela plataforma duas possibilidades para o fazer, sendo elas através de um menu ou de uma barra lateral.

4.3.3 Barra Lateral/Dock

Quando os dispositivos são reconhecidos através da API e criados os componentes, esses componentes são registados também na barra lateral. Nesta barra lateral o utilizador tem apresentados todos os dispositivos ao seu dispôr. Para adicionar os componentes ao editor tem dois tipos de interação possível, sendo eles o **clique** e o **drag and drop**.

Esta barra lateral é incorporada no editor através da utilização de um plugin chamado *rete-dock-plugin*. Na figura 4.5 podemos ver como se instancia o plugin responsável pela barra lateral

```
editor.use(DockPlugin.default, {
  container:document.querySelector('.dock'),
  itemClass: 'dock-item', // default: dock-item
  plugins: [VueRenderPlugin] // render plugins
});
```

Figura 4.5: Utilização do plugin *rete-dock-plugin* para criação da barra lateral

4.3.4 Menu

A outra possibilidade oferecida aos utilizadores é um menu que apresenta de forma mais organizada todos os dispositivos. Além dos dispositivos estarem agrupados por grupo do Home Assistant, este menu também oferece uma barra de procura para permitir ao utilizador um acesso mais direto ao componente pretendido. A configuração deste plugin no editor pode ser consultada na figura 4.6.

Na figura 4.7 podemos observar as duas opções distintas que a plataforma oferece para a adição de componentes ao editor. À direita temos a barra lateral e no editor podemos observar o menu que se encontra organizado por grupos.

```
editor.use(ContextMenuPlugin, {
    searchBar: true,
    delay: 100,
    allocate(component) {
        groups.sort()
        for (var i=0;i<groups.length;i++){
          if(component.name=="Time Clock"){
            return ['Others'];
          }else{
            if (getDevice(component.name).split(".")[0]==groups[i]){
              return[capitalize(groups[i])]
            }
          }
        }
        return ['Others'];
        function capitalize(string)
        {
            return string.charAt(0).toUpperCase() + string.slice(1);
        }
      },
    rename(component) {
        return component.name;
    }
});
```

Figura 4.6: Configuração do plugin para criação do menu

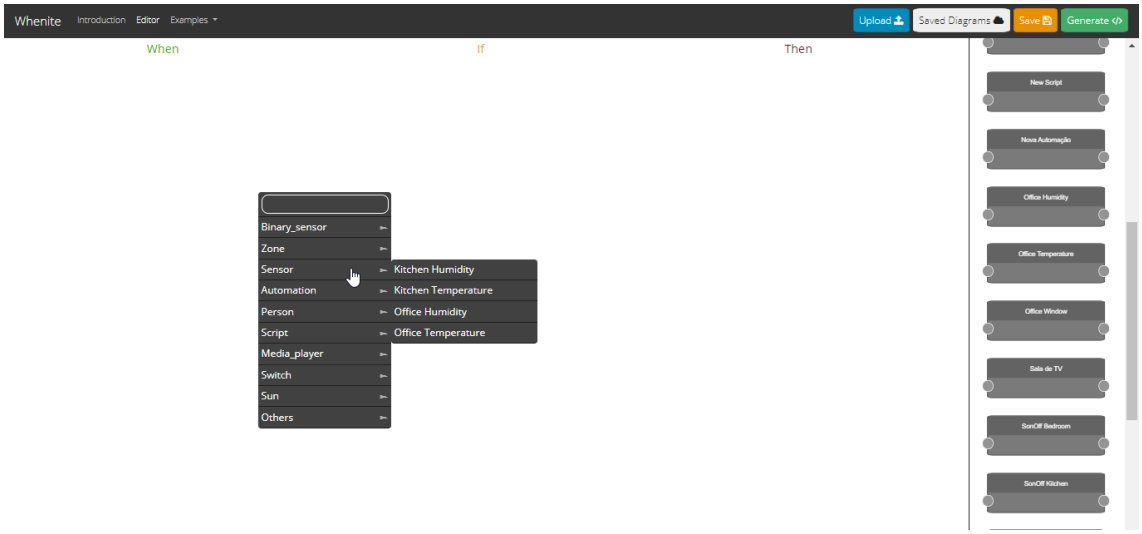


Figura 4.7: Possibilidades para adição de componentes ao editor.

4.3.5 Visualização e acompanhamento de automações

Para o utilizador poder acompanhar a sua automação de forma a perceber se o que vai ser compilado é exatamente o que ele pretende, foi criado uma zona da plataforma em que a automação que está a ser criada é apresentada no formato de *When If Then*. Na figura 4.8 podemos observar um exemplo de uma automação e a pré-visualização correspondente.

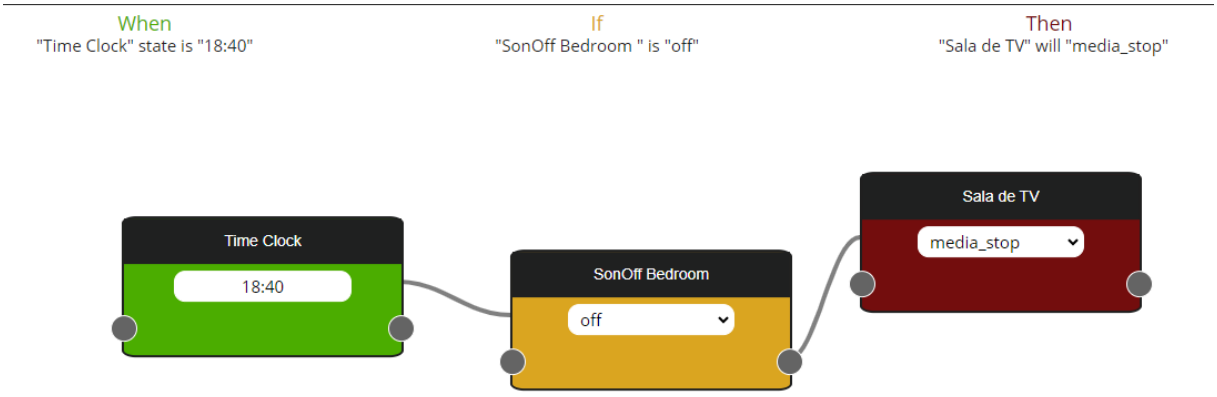


Figura 4.8: Zona responsável pela pré-visualização da automação que está a ser criada

Sempre que é feita uma ligação no editor, existe uma função que é responsável por fazer a coloração de cada componente consoante o grupo ao qual pertence. O código de cores usado foi o verde, amarelo e vermelho.

- **Verde**: O verde é utilizado para representar os *triggers* que são responsáveis por encadear o conjunto de ações pretendidas.
- **Amarelo**: Foi usado o amarelo para representar as condições.
- **Vermelho**: O vermelhor é utilizado para a representação de dos componentes que pertencem ao grupo das ações.

Esta coloração dos componentes permite ao utilizador uma identificação mais rápida do grupo a que pertence cada um dos componentes que tem ligados na automação que está a ser criada.

4.4 Armazenamento local

Sempre que o utilizador pretender guardar uma automação, é o Armazenamento Local do browser [71] o recurso utilizado para executar essa função. Quando uma automação é guardada a plataforma regista essa automação no Armazenamento Local e posteriormente é possível aceder às automações guardadas e o utilizador tem a possibilidade de fazer download da automação pretendida para o seu computador.

Podemos observar na figura 4.9 como é que as automações guardadas são apresentadas ao utilizador.

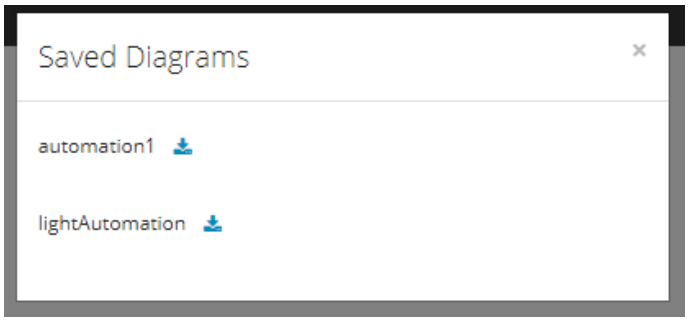


Figura 4.9: Apresentação das automações guardadas pelo utilizador

4.4.1 Formato do diagrama guardado

A framework Rete.js possibilita a exportação do diagrama desenhado no editor. Quando o utilizador pretende fazer o download do diagrama da automação para o seu computador é colocado para um ficheiro JSON toda a informação relativa ao diagrama. Na figura 4.10 temos uma parte de um ficheiro que corresponde a uma automação criada na plataforma.

Neste ficheiro JSON, cada nó é representado com um identificador único e são apresentadas todas as ligações que esse nó apresente com outros, indicando se as mesmas existem na saída ou na entrada do nó em questão.

4.5 Compilador

O compilador é a parte mais importante da plataforma pois é o compilador que vai interpretar o diagrama e gerar o código YAML da automação. O código depois de construído é apresentado ao utilizador para que seja possível que o mesmo copie o código e possa adicioná-lo ao ficheiro de automações do seu Home Assistant.

Quando o utilizador pretende compilar o diagrama é-lhe pedido que indique um nome e uma descrição para a sua automação e, após clicar no botão de gerar, é-lhe apresentado o respetivo código. Na figura 4.11 pode-se observar o referido acima. À esquerda temos um pequeno formulário onde o utilizado é questionado sobre o nome e a descrição da sua automação e à direita podemos observar um exemplo do código YAML para uma automação.

```
{
    "id":"demo@0.1.0",
    "nodes":{
        "20":{
            "id":20,
            "data":{
                "data":">25"
            },
            "inputs":{
                "inp":{
                    "connections":[]
                }},
            "outputs":{
                "out":{
                    "connections":[{
                            "node":22,
                            "input":"inp",
                            "data":{}
                        }]}},
            "position":[
                -405,
                -122.6875
            ],
            "name":"Kitchen Temperature"
        },
        "22":{
            "id":22,
            "data":{
                "data":"on"
            },
            "inputs":{
```


Figura 4.10: Código JSON representativo de um diagrama criado na plataforma

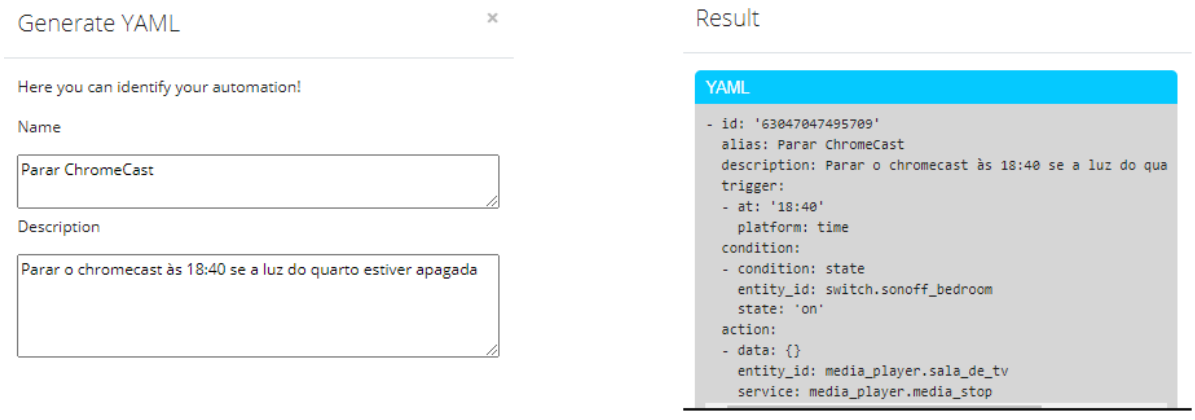


Figura 4.11: Menu responsável por apresentação de código compilado

Capítulo 5

Resultados

Ao longo deste capítulo, numa primeira fase será apresentada a abordagem escolhida para o tratamento de erros/bugs que a plataforma sofra e numa segunda fase será apresentados em primeiro lugar um questionário que contém um teste de usabilidade e uma seção de feedback e de seguida apresentadas estatísticas de utilização da mesma através da análise de *logs* provenientes no nginx.

5.1 Tratamento de Erros

Ao colocar o produto na fase de produção inúmeros fatores têm de ser considerados, entre os quais se encontram os erros, talvez o fator mais importante a ser considerado. Os erros precisam de passar por várias fases entre as quais se encontram a deteção do erro e a resolução do mesmo.

Numa primeira fase, é necessário detetar o erro e registá-lo para consulta. Para este fim, como falado no capítulo 2 existem plataformas que têm como função captar possíveis erros que possam surgir numa aplicação e armazená-los.

Para fazer a monitorização da plataforma criada foi escolhido o Sentry uma vez que, como foi possível constatar no capítulo referido, é a plataforma mais utilizada atualmente que permite efetuar o controlo dos erros. Foram configuradas as notificações no Slack e as notificações por email o que é muito útil uma vez que, quando surge um novo erro ou algum problema com a aplicação, é lançada uma notificação automaticamente.

Na figura 5.1 podemos observar exemplos das notificações de erro que são recebidas pelo Slack e pelo email, à esquerda e direita respetivamente. Estas notificações, além de apresentarem o tipo de erro, apresenta também a zona do código na qual esse erro foi originado, tornando mais fácil a perceção e posterior resolução do mesmo.

Além de configuradas as notificações para o email e para o Slack, o Sentry envia relatórios semanais com o número de erros que surgiram na plataforma para fins estatísticos, como podemos observar na figura 5.2.

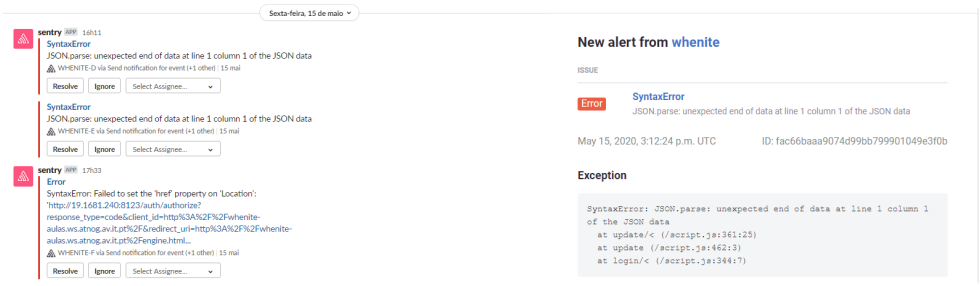


Figura 5.1: Notificação de um erro enviado para o Slack (esquerda) e para o email (direita).

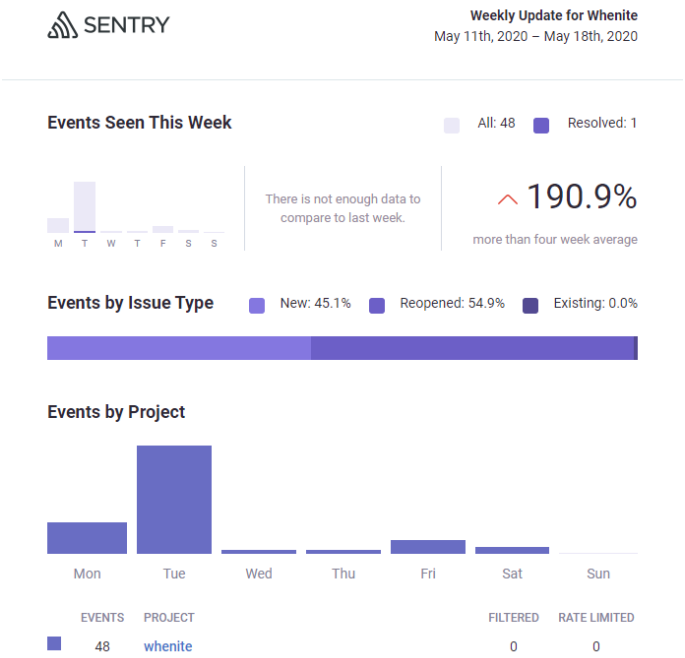


Figura 5.2: Relatório da semana de 11 a 18 de maio enviado pelo Sentry

Após o erro ter sido detetado, é necessário resolvê-lo e, para isso, o Sentry também oferece uma ajuda. É possível através do Sentry, atribuir o erro a uma pessoa específica para que essa pessoa o resolva o mais rapidamente possível. Uma vez resolvido o erro, a plataforma é de novo lançada para utilização sendo que o Sentry continua configurado para detetar o surgimento de novos erros.

De modo a simplificar a explicação deste processo de tratamento de erros, podemos observar como se procede todo este processo na figura 5.3.

Como se pode observar no diagrama, podem-se identificar 3 fases no processo de tratamento de erros. A primeira fase, como explicada anteriormente passa pela deteção do erro. Após ser detetado o erro e registado pela plataforma, o Sentry, começa a segunda fase, onde o erro é atribuído a alguém para ser resolvido. Após a resolução do erro estar concluída, a

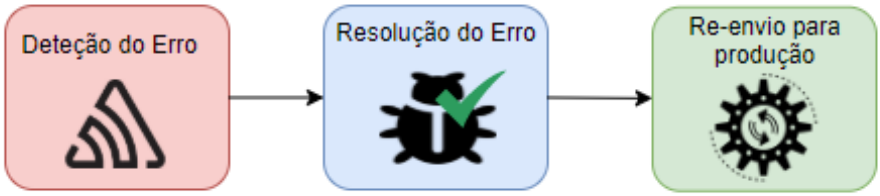


Figura 5.3: Diagrama explicativo do processo de tratamento dos erros

plataforma é lançada novamente para a fase de produção, onde continua a ser monitorizada pelo Sentry.

5.1.1 Erros detetados

Na figura 5.4 podemos observar os tipos de erros que o Sentry detetou nas utilizações dos utilizadores que testaram a plataforma e que posteriormente foram resolvidos.

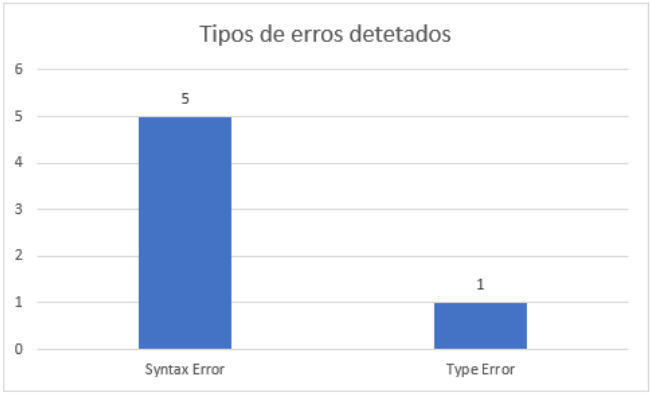


Figura 5.4: Erros detetados e posteriormente resolvidos, agrupados por tipo

Como podemos concluir observando a figura 5.5, retirada da página do Sentry que apresenta estatísticas relativas ao surgimento de erros da plataforma, a maioria dos erros detetados surgiram por volta da segunda semana do mês de maio de 2020, momento em que a plataforma foi lançada a primeira vez. Analisando o correspondente ao mês de junho, pode-se observar que já existe uma menor quantidade de erros detetados.

5.2 Questionário

Para validar a plataforma foi realizado um questionário composto por duas partes distintas. A primeira parte consistia num teste de usabilidade, em que era pedido aos utilizadores

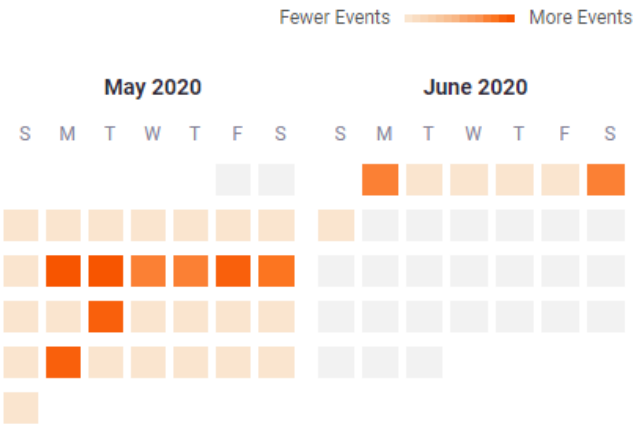


Figura 5.5: Evolução do número de erros detetados durante os meses de maio e junho de 2020

que realizassem um conjunto de tarefas e indicassem a dificuldade que sentiram ao realizar as mesmas. Na segunda parte do questionário é pedido aos utilizadores algumas opiniões pessoais acerca da plataforma e é facultada uma zona de feedback onde os utilizadores podem recomendar algumas alterações que achem relevantes para melhorar a plataforma.

Os questionários foram disponibilizados a nível nacional no fórum e no canal do Discord da comunidade portuguesa de Home Assistant e a nível mundial no fórum da comunidade do Home Assistant global. Os inquiridos são utilizadores recorrentes no tema do Home Assistant e das automações, pelo que seriam os mais indicados para o teste da plataforma. O questionário esteve recetivo a respostas durante 20 dias e no total foram respondidos um total de 21 questionários. Apesar das poucas respostas obtidas, também foi recebido feedback nos canais do Discord e no fórum das comunidades mencionadas que foi ajudando a melhorar a plataforma.

5.2.1 Teste de usabilidade

Como dito acima, foi pedido aos utilizadores que testaram a plataforma que realizassem algumas tarefas e indicassem a dificuldade sentida na sua realização.

A dificuldade seria classificada de 1 a 5 sendo que o nível 1 representaria que a tarefa proposta foi muito díficil de realizar e o nível 5 indicaria que a tarefa foi muito fácil de realizar.

De seguida serão apresentadas algumas das tarefas que foram apresentadas aos utilizadores e serão apresentados gráficos com a dificuldade sentida pelos mesmos para a resolução da tarefa.

Na primeira tarefa era pedido que os utilizadores iniciassem sessão na plataforma com a sua instância do Home Assistant.

Podemos concluir através da figura 5.6 que todos os intervenientes acharam que a rea-

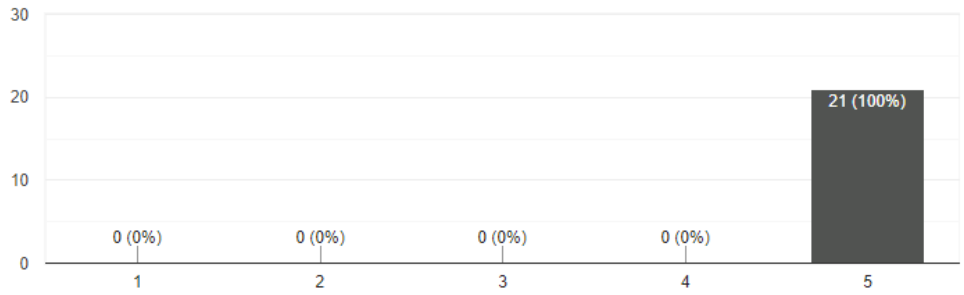


Figura 5.6: Respostas relativas à tarefa de iniciar sessão com o Home Assistant

lização desta tarefa era de muito fácil execução.

Na segunda e terceira tarefas era pedido que fossem colocados os dispositivos pretendidos para a automação no editor, pedido aos utilizadores que tentassem utilizar as duas possibilidades que a plataforma oferece para a inserção de dispositivos. A terceira tarefa pedia ao utilizador para colocar um determinado tipo de dispositivo para levar o utilizador a utilizar o menu de inserção, uma vez que nesse menu os dispositivos estão agrupados por domínio do Home Assistant..

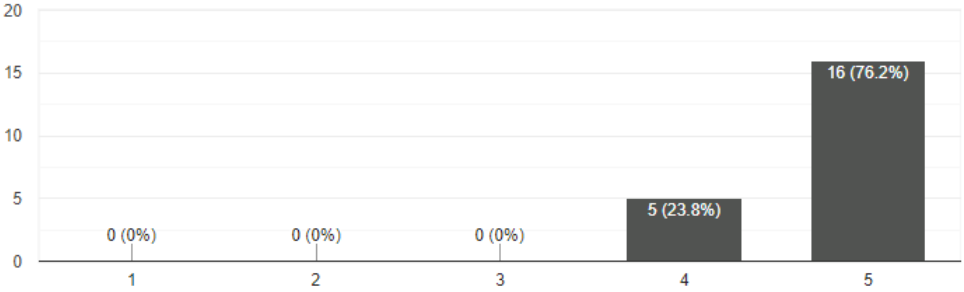


Figura 5.7: Respostas relativas à tarefa de adicionar os dispositivos no editor

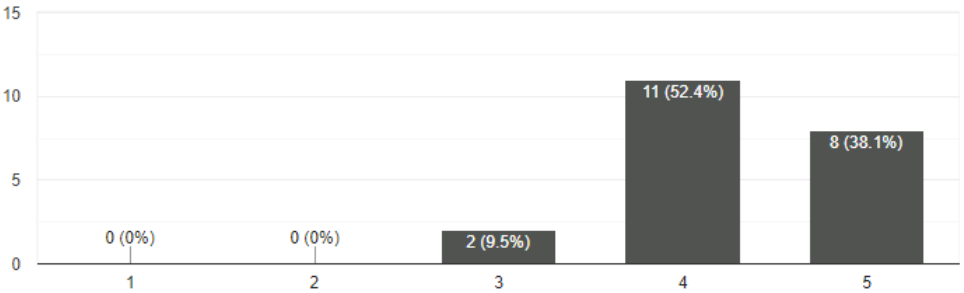


Figura 5.8: Respostas relativas à tarefa que pedia para adicionar um tipo de dispositivo específico

Através da análise dos gráficos 5.7 e 5.8, podemos concluir que ambas as tarefas foram

resolvidas com alguma facilidade, à exceção da terceira tarefa que obteve duas respostas no nível três. Estas respostas podem dever-se ao facto do menu de inserção apenas ser aberto com o clique direito do rato, o que pode não ser muito intuitivo.

Na quarta tarefa era pedido para estabelecer as ligações entre os dispositivos de modo a criar um fluxo que pudesse ser convertido numa automação. A tarefa cinco perguntava ao utilizador se a lógica da automação que estavam a criar era bem apresentada por parte do visualizador que foi adicionado como descrito no subcapítulo 4.3.5.

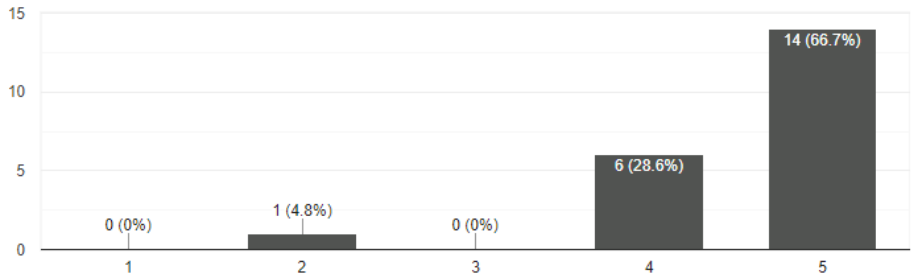


Figura 5.9: Respostas relativas à tarefa de criar um fluxo no diagrama

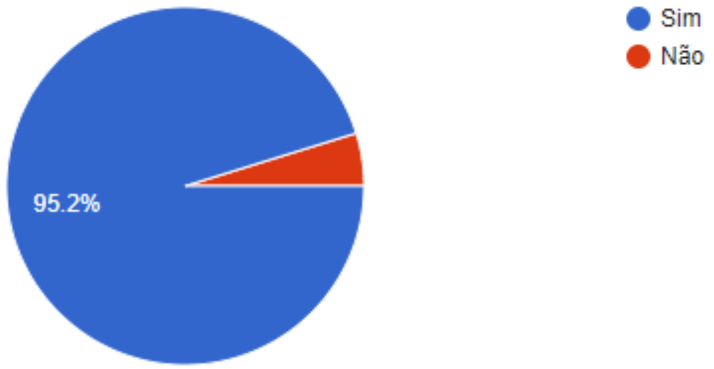


Figura 5.10: Respostas à pergunta se a plataforma apresentava bem previsto o fluxo criado

Nas figuras 5.9 e 5.10 podemos constatar que na generalidade as tarefas propostas também foram de fácil execução à exceção de um utilizador (4.8% dos inquiridos) que apontou que teve alguma dificuldade em efetuar as ligações entre os dispositivos e tambem indicando que a plataforma não apresentava corretamente uma pré-visualização da automação como ele teria desenhado.

A tarefa seis remetia para um dos principais objectivos da plataforma, a geração do código YAML correspondente à automação criada.

Na figura 5.11 podemos observar o gráfico onde é apresentada a dificuldade sentida na geração do código YAML. Na generalidade a tarefa foi realizada com facilidade. Houve uma

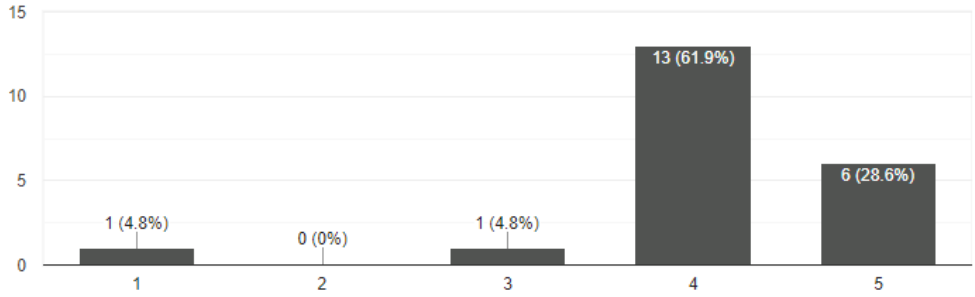


Figura 5.11: Respostas relativas à tarefa de obter o código compilado do diagrama

resposta que indicou que a dificuldade da tarefa era elevada. Esta resposta pode-se dever a um botão pouco intuitivo, o que levou a uma perceção mais lenta por parte do utilizador.

A sétima tarefa tinha como objetivo guardar a automação que estava a ser criada por parte do utilizador. No seguimento da sétima tarefa, surge uma pergunta ao utilizador que pergunta se a automação que tinha acabado de guardar se encontrava na lista de automações guardadas.

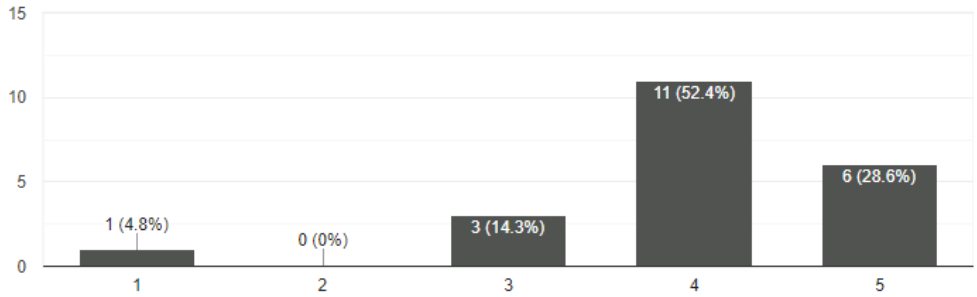


Figura 5.12: Respostas relativas à tarefa de guardar a automação

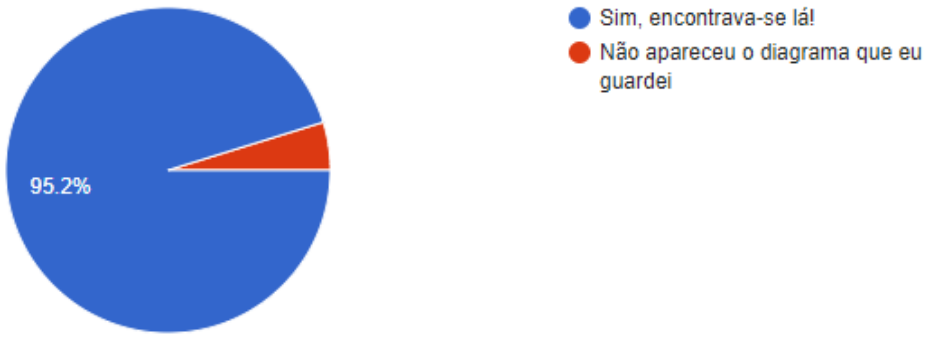


Figura 5.13: Respostas à pergunta que constatava que o diagrama era guardado corretamente

Analisando os gráficos 5.12 e 5.13 podemos concluir que a realização desta tarefa teve uma dificuldade muito reduzida. No gráfico 5.13 observamos que em 4.8 dos inquiridos, o diagrama que o utilizador guardou não apareceu na lista. Uma vez que os diagramas são guardados no *Local Storage* do browser, consoante o browser em que estamos a utilizar a plataforma ou as configurações do mesmo, podem ser justificação para que não tenha sido possível observar o diagrama guardado.

Na tarefa nove pede-se ao utilizador que efetue o download de uma automação que se encontre guardada e nesse seguimento carregue essa automação para o editor novamente e indicar se o diagrama da automação é o mesmo ou se houve algum problema.

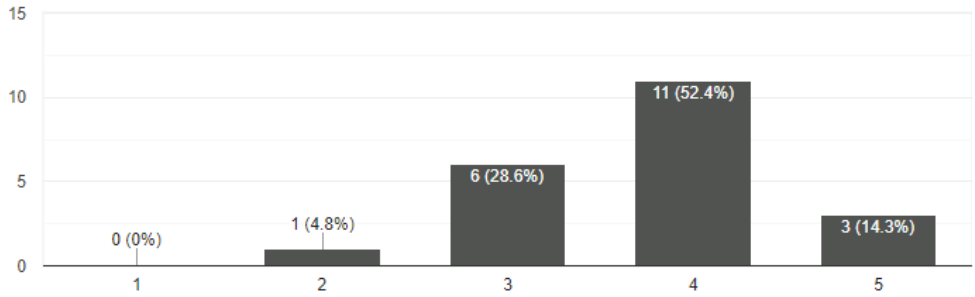


Figura 5.14: Respostas relativas à tarefa de efetuar o download do diagrama

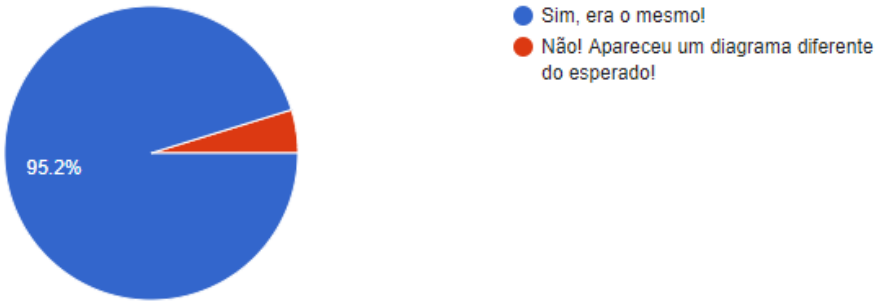


Figura 5.15: Respostsa à pergunta se após o upload da automação se mantinha tudo como esperado

Através da análise do gráfico da figura 5.14 podemos observar que não houve muita dificuldade por parte dos utilizadores na realização desta tarefa e, nesse seguimento, na figura 5.15 constatamos que 95.2% dos diagramas que foram transferidos para o computador e importados novamente não sofreram qualquer alteração. Nos 4.8% que indicam que o diagrama não foi o mesmo que transferiram para o seu computador pode ter acontecido algum erro que levou a que o diagrama sofresse algumas alterações.

Como foi possível observar ao longo deste subcapítulo, algumas tarefas tiveram algumas discordâncias como o que era suposto mas, nesses casos, se a origem desses problemas foi

um erro de código ou similar, a plataforma de monitorização de erros foi alertada com um possível erro que possa ter surgido e posteriormente esse erro será resolvido.

5.2.2 Feedback

A segunda parte do questionário, como dito anteriormente, é uma secção onde são realizadas algumas perguntas ao utilizador e é pedido uma opinião de cada um acerca da plataforma.

A primeira pergunta feita perguntava aos utilizadores se já tinham utilizado alguma vez uma plataforma externa para a criação de automações. Na figura 5.16 termos apresentadas as respostas a esta pergunta.

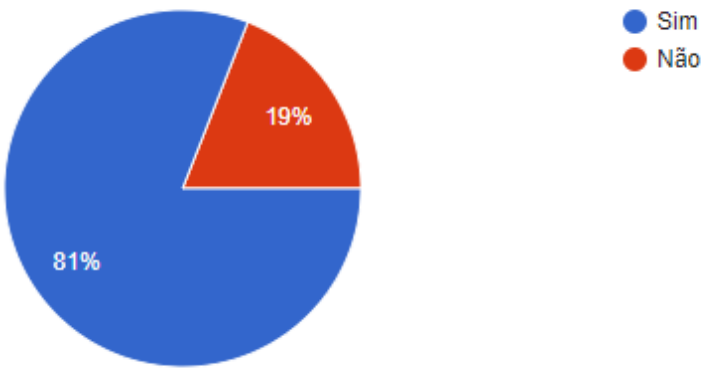


Figura 5.16: Respostas à pergunta se já utilizou alguma plataforma externa para criar automações

Podemos observar na figura 5.16 que 81% dos inquirido já tinha usado alguma plataforma para a criação de automações ao contrário dos restantes 19% que nunca tinham utilizado nenhuma plataforma. No seguimento desta pergunta foi questionado, no caso de ter respondido "sim", que plataformas é que utilizaram. De todos os utilizadores que responderam sim à questão, todos já tinha utilizado o Node-Red como plataforma para criação de automações, sendo que um também referenciou já ter utilizado o Particle.

Outra das perguntas apresentadas questionava os utilizadores se utilizariam o Whenite para a criação das suas automações. Na figura 5.17 são apresentadas as respostas a essa pergunta.

Na zona de opiniões, foi pedido aos utilizadores que indicassem possíveis melhorias para a plataforma. Algumas respostas foram direcionadas ao aspeto estético da plataforma referenciando que precisaria de umas melhorias nesse aspeto. Também foi indicado por alguns inquiridos que poderia ter suporte para a língua portuguesa, uma vez que a plataforma foi toda concebida em inglês. Foi sugerido por um utilizador que era interessante que fosse possível ter uma pré-visualização de automações que já tivessem sido guardadas.

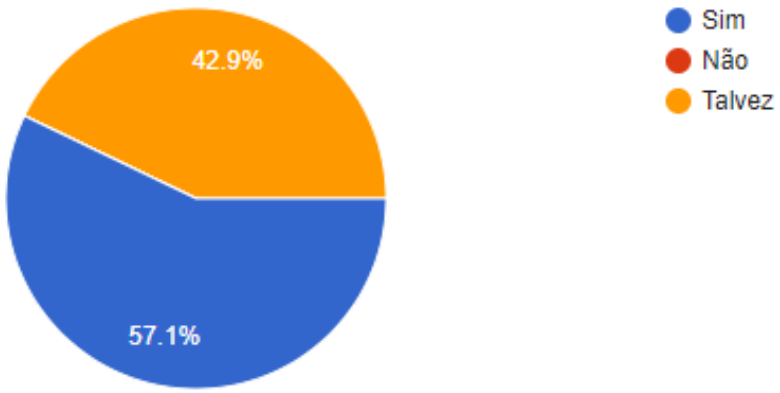


Figura 5.17: Resposta à pergunta se os utilizadores utilizariam o Whenite

5.3 Análise dos *logs* do nginx

Para disponibilizar a plataforma concebida para os utilizadores, foi utilizada uma máquina virtual no Instituto de Telecomunicações da Universidade de Aveiro onde foi instalado o nginx como *webserver*. O nginx registou todos os acessos que foram realizados à plataforma e, através desses registos, foi possível retirar algumas estatísticas de utilização que serão apresentadas de seguida.

Na figura 5.18 temos apresentados o número de visitantes que o Whenite teve durante o período de 11 de maio a 31 de maio de 2020. Podemos concluir que ao longo do tempo existe uma utilização regular da plataforma. No gráfico apresentado na figura 5.19 temos apresentados os browsers utilizados pelos utilizadores que acederam ao Whenite. Pode-se reparar num claro domínio por parte do Chrome, que é atualmente um dos browsers mais utilizados. O browser utilizado pode ser determinante da identificação de erros, uma vez que esta plataforma funciona na web.

À semelhança dos gráficos anteriores, nas figuras 5.20 e 5.21 apresentamos as mesmas estatísticas mas desta vez para o período de 1 a 5 de junho do ano de 2020.

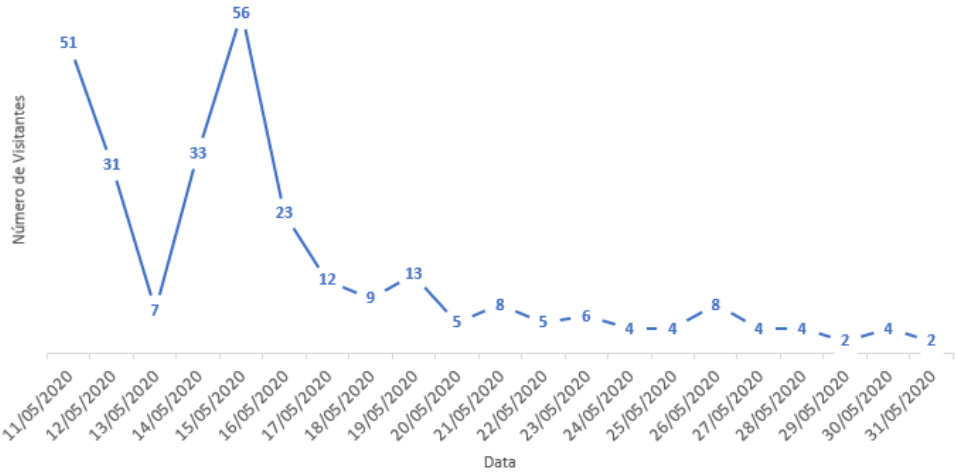


Figura 5.18: Gráfico do número de visitas ao longo do mês de maio

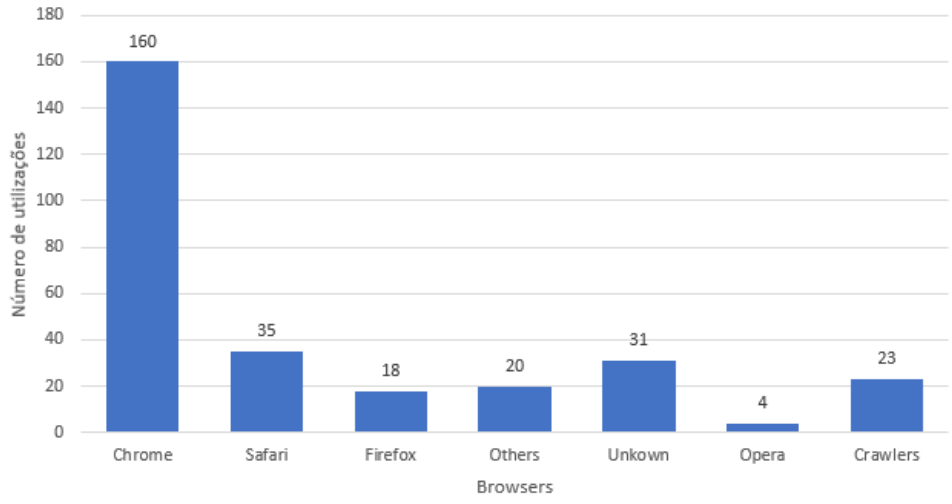


Figura 5.19: Gráfico do utilizações dos browsers utilizados

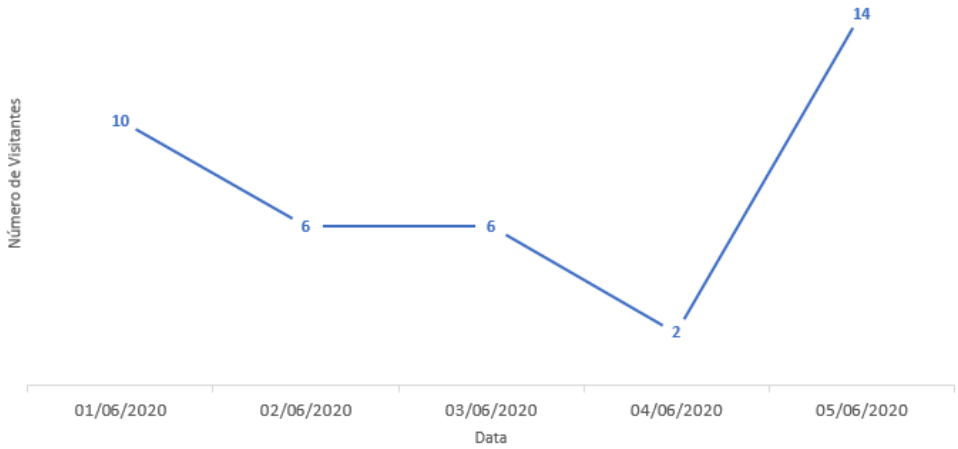


Figura 5.20: Gráfico do número de visitas ao longo do mês de junho

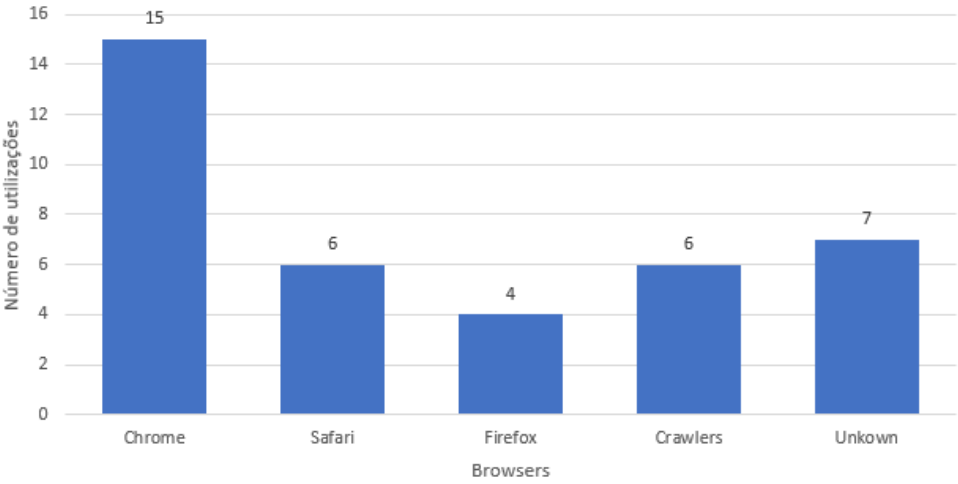


Figura 5.21: Gráfico do utilizações dos Browsers utilizados

Capítulo 6

Conclusões

Esta dissertação teve como principal objetivo a criação de uma nova plataforma que pudesse concorrer com as atuais plataformas existentes no mercado no que diz respeito a facilitar a criação de automações por parte dos utilizadores.

A nova solução tinha como objectivos principais a facilidade de acesso e a pouca intervenção do utilizador, de modo a ser utilizada por todos os tipos de utilizadores com diferentes graus de conhecimento.

Inicialmente foi realizado uma pesquisa sobre as temáticas da área, nomeadamente ao nível dos assistentes e de como cada um tentava facultar ao utilizador a possibilidade de automatizar a sua habitação. Foram investigados vários assistentes desde o Home Assistant, o openHAB e os assistentes pessoais com capacidade de controlar as habitações, no caso o Google Home/Assistant e o Amazon Alexa.

Foram estudadas diversas plataformas externas às quais os utilizadores recorriam de modo a facilitar a criação de automações como é o caso do Node-Red e do Particle. Foi também estudada a plataforma do Thingsboard que para além de funcionar como assistente também possui um editor próprio que ajuda os utilizadores na criação das automações. Foi possível ao longo de um estudo e teste destas plataformas verificar alguns aspetos que não estariam de todo bem abordados, nomeadamente ao nível da dificuldade de configuração dos componentes e da necessidade de conhecimentos por parte dos utilizadores e, sendo assim, pensar numa solução que pudesse solucionar esses problemas.

Além da pesquisa sobre as plataformas às quais os utilizadores recorriam, foi feita uma pesquisa também na área da monitorização de erros, uma vez que caso algum erro surgisse na plataforma, não ser preciso o envolvimento do utilizador no processo de reportar esses erros. Essas plataformas são configuradas no código fonte da aplicação e sempre que um erro acontece é imediatamente reportado para a plataforma e junto com ele alguns dados e estatísticas que possam ser relevantes para a resolução do erro.

A próxima fase passou por pensar e desenhar uma arquitetura que fosse o mais modular possível para que no futuro se pudessem fazer alterações sem comprometer a arquitetura base

da plataforma.

Após ter uma arquitetura base bem pensada, foi começado um estudo de frameworks que pudessem ser utilizadas para desenvolver esta nova plataforma. A programação em fluxos foi um dos primeiros requisitos que foram achados de valor uma vez que a criação do fluxo ajuda numa melhor perceção da automação. Foram testadas algumas frameworks mas a selecionada acabou por ser Rete.js, uma framework muito flexível e modular que cumpriria com todas as necessidades.

Através da adição de plugins facultados pela framework foi possível oferecer ao utilizador maior facilidade na execução de algumas tarefas no editor desenvolvido.

Para validar este novo conceito criado foram utilizadas duas abordagens distintas. A primeira abordagem passava pela realização de um questionário que continha teste de usabilidade à plataforma e uma zona de feedback. Apartir deste questionário foi possível constatar que na generalidade os utilizadores que testaram a plataforma acharam o conceito muito interessante e a usabilidade da plataforma estava muito boa. Na zona de feedback os utilizadores deram dicas sobre possíveis aspetos que poderiam ser melhorados na plataforma.

A segunda abordagem passou pela análise dos *logs* do nginx, o webserver instalado para facultar o acesso da plataforma a todos os utilizadores. Através da análise dos seus registos foi observado que apesar de ser um novo conceito e uma plataforma nova no mercado, ao longo do tempo foi sendo usada constantemente por alguns utilizadores o que pode fazer-se concluir que alguns utilizadores recorrem à plataforma para a criação das suas automações.

Juntando os resultados das duas abordagens de validação pode-se concluir que a plataforma foi bem aceite pela comunidade.

Referências

[1] Pedro José Santos Monteiro. Aplicação Android para sistema de Domótica. page 117, 2015. Instituto Politécnico de Viseu.

[2] Rafael Almeida Costa. Casa Inteligente com recurso a tecnologias OpenSource. 2018. Instituto Politécnico de Viseu.

[3] Auriza Lopes De Barros. Edifícios Inteligentes e a Domótica Proposta de um Projecto de Automação Residencial. 2010. Universidade Jean Piaget de Cabo Verde.

[4] Renato Jorge and Caleira Nunes. Análise Comparativa de Tecnologias para Domótica. Technical report.

[5] Niko Brasseur. Smart Homes & Domotica - A-knowledge. `https://www.a-knowledge.eu/a-knowledge/smart-homes-domotica/`. (Visitado em 09/12/2019).

[6] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, oct 2010.

[7] Daniel Giusto, Antonio Iera, Giacomo Morabito, and Luigi Atzori. *The Internet of Things*. Springer New York, New York, NY, 2010.

[8] Sinha G Gaurav. The Evolution of Smart Home Technology, 2018. `http://blog.bccresearch.com/the-evolution-of-smart-home-technology/`. (Visitado em 19/12/2019).

[9] Drew Hendricks. The History of Smart Homes, 2014. `https://www.iotevolutionworld.com/m2m/articles/376816-history-smart-homes.htm`. (Visitado em 19/12/2020).

[10] João Ferreira. *Interface homem-máquina para domótica baseado em tecnologias Web*. PhD thesis, Faculdade de Engenharia da Universidade do Porto, 2008.

[11] Miguel Jorge Monteiro de Magalhães Ferreira and Eduardo José Freitas Castro Lopes. *Sistemas Domóticos*, volume 53. 2019.

[12] José Augusto Alves and José Mota. *Casas Inteligentes*. Centro Atlântico, 2003.

[13] Pedro Gouveia. *Automação em Ambientes Residenciais, Universidade de Aveiro*. PhD thesis, Universidade de Aveiro, 2009.

[14] A. J.Bernheim Brush, Bongshin Lee, Ratul Mahajan, Sharad Agarwal, Stefan Saroiu, and Colin Dixon. Home automation in the wild: Challenges and opportunities. *Conference on Human Factors in Computing Systems - Proceedings*, (May):2115–2124, 2011.

[15] Amazon.com: Amazon Echo & Alexa Devices: Amazon Devices & Accessories, 2019. `https://www.amazon.com/Amazon-Echo-And-Alexa-Devices/b?ie=UTF8{&}node=9818047011`. (Visitado em 14/05/2020).

[16] Irene Lopatovska, Katrina Rink, Ian Knight, Kieran Raines, Kevin Cosenza, Harriet Williams, Perachya Sorsche, David Hirsch, Qi Li, and Adrianna Martinez. Talk to me: Exploring user interactions with the Amazon Alexa. *Journal of Librarianship and Information Science*, 51(4):984–997, dec 2019.

[17] Hyunji Chung, Jungheum Park, and Sangjin Lee. Digital forensic approaches for Amazon Alexa ecosystem. In *DFRWS 2017 USA - Proceedings of the 17th Annual DFRWS USA*, volume 22, pages S15–S25. Digital Forensic Research Workshop, aug 2017.

[18] DEAN BRYEN. Amazon alexa and the alexa skills kit. 2016. `https://www.slideshare.net/AmazonWebServices/please-meet-amazon-alexa-and-the-alexa-skills-kit`. (Visitado em 14/05/2020).

[19] David Ludlow. How to make Amazon Alexa Routines – smart home automation made easy — Trusted Reviews, 2019. `https://www.trustedreviews.com/how-to/how-to-make-alexa-routines-smart-home-automation-made-easy-3426061`. (Visitado em 14/05/2020).

[20] Jared Newman. How to use Alexa Routines to make your Amazon Echo even smarter — TechHive, 2018. `https://www.techhive.com/article/3327501/how-to-use-alexa-routines.html`. (Visitado em 14/05/2020).

[21] Daniel Meyers. Google Assistant Smart Home Part 2: API Implementation, 2019. `https://medium.com/google-developers/jdanielmyers-smart-home-eac8f87fd56`. (Visitado em 14/05/2020).

[22] Mitja Rutnik. Google Assistant routines — what are they and how to set them up?, 2020. `https://www.androidauthority.com/google-assistant-routines-2-867959/`. (Visitado em 18/05/2020).

[23] Frontend of home assistant - home assistant. https://www.home-assistant.io/docs/frontend/. (Visitado em 16/12/2019).

[24] Daniel Costa. Home Assistant – O seu assistente pessoal na domótica! – Lógica da Mecatrónica. https://www.logicamecatronica.com/2017/01/04/home-assistant-o-seu-assistente-pessoal-na-domotica/. (Visitado em 16/12/2019).

[25] Architecture — Home Assistant Developer Documentation. https://developers.home-assistant.io/docs/architecture_index/. (Visitado em 12/05/2020).

[26] Integrations - Home Assistant. https://www.home-assistant.io/integrations/. (Visitado em 12/05/2020).

[27] Introduction — openHAB. https://www.openhab.org/docs/. (Visitado em 12/05/2020).

[28] Kevin Arrows. Home Assistant Vs OpenHAB - Appuals.com, 2019. https://appuals.com/home-assistant-vs-openhab/. (Visitado em 12/05/2020).

[29] Dashboard ui suitable for tablets - setup, configuration and use / items & sitemaps - openhab community. https://https://community.openhab.org/t/dashboard-ui-suitable-for-tablets/2329. (Visitado em 12/05/2020).

[30] Florian Heimgaertner, Stefan Hettich, Oliver Kohlbacher, and Michael Menth. Scaling home automation to public buildings: A distributed multiuser setup for OpenHAB 2. In *GIoTS 2017 - Global Internet of Things Summit, Proceedings*. Institute of Electrical and Electronics Engineers Inc., aug 2017.

[31] Equinox — The Eclipse Foundation, 2018. https://www.eclipse.org/equinox/framework/. (Visitado em 12/05/2020).

[32] Apache Karaf - Enterprise runtime everywhere. https://karaf.apache.org/. (Visitado em 12/05/2020).

[33] Home · openhab/openhab1-addons Wiki. https://github.com/openhab/openhab1-addons/wiki. (Visitado em 12/05/2020).

[34] Building_iot_systems_with_openhab.pdf. https://elinux.org/images/0/0a/Building_IoT_systems_with_openHAB.pdf. (Visitado em 12/05/2020).

[35] Domoticz. https://www.domoticz.com/. (Visitado em 12/05/2020).

[36] Raspberry Pi automation or Homey? Compare Domoticz, Home Assistant and Homey — Homey. https://homey.app/en-ie/wiki/

raspberry-pi-automation-or-homey-compare-domoticz-home-assistant-homey/. (Visitado em 12/05/2020).

[37] Blockly — Google Developers. https://developers.google.com/blockly. (Visitado em 07/02/2020).

[38] Automation - domoticz. https://www.domoticz.com/wiki/Automation. (Visitado em 12/05/2020).

[39] IFTTT: Every thing works better together. https://ifttt.com/. (Visitado em 07/06/2020).

[40] Elize Betters. O que é o IFTTT e como ele funciona?, 2018. https://www.pocket-lint.com/pt-br/casa-inteligente/noticias/130082-o-que-e-ifttt-e-como-funciona. (Visitado em 07/06/2020).

[41] Chen Liu, Xu Peng Dong, and Zheng Qiu Yang. Research of modern enterprise intelligent system based on rule engine and workflow. *Proceedings - 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, ICIS 2010*, 1:594–597, 2010.

[42] Bertha Mazon-Olivo, Dixys Hernández-Rojas, José Maza-Salinas, and Alberto Pan. Rules engine and complex event processor in the context of internet of things for precision agriculture. *Computers and Electronics in Agriculture*, 154:347–360, nov 2018.

[43] Kris Wouk. What Is a Rules Engine for IoT? - IoT Tech Trends, 2019. https://www.iottechtrends.com/rules-engine-for-iot/. (Visitado em 16/12/2019).

[44] Nhan Bao and Sun Tae Chung. A rule-based smart thermostat. In *ACM International Conference Proceeding Series*, pages 20–25. Association for Computing Machinery, nov 2018.

[45] Seven Automation and Technologies For. A guide to : Rules Engines.

[46] Node-RED. https://nodered.org/. (Visitado em 16/12/2019).

[47] Conor Allison. How to use Google Home and Assistant as a smart alarm clock, 2020. https://www.the-ambient.com/how-to/use-google-home-assistant-alarm-clock-bedtime-music-545. (Visitado em 04/06/2020).

[48] V. Chayapathy, G. S. Anitha, and B. Sharath. IOT based home automation by using personal assistant. In *Proceedings of the 2017 International Conference On Smart Technology for Smart Nation, SmartTechCon 2017*, pages 385–389. Institute of Electrical and Electronics Engineers Inc., may 2018.

[49] Stephen A White. Introduction to BPMN. Technical report, 2004.

[50] What is Business Process Modeling Notation — Lucidchart. https://www.lucidchart.com/pages/bpmn. (Visitado em 07/06/2020).

[51] Complex Event Processing: An Overview with Use Cases — Hazelcast. https://hazelcast.com/glossary/complex-event-processing/. (Visitado em 07/06/2020).

[52] What is Complex Event Processing? - Databricks. https://databricks.com/glossary/complex-event-processing. (Visitado em 07/06/2020).

[53] Nam Ky Giang, Michael Blackstock, Rodger Lea, and Victor C.M. Leung. Developing IoT applications in the Fog: A Distributed Dataflow approach. In *Proceedings - 2015 5th International Conference on the Internet of Things, IoT 2015*, pages 155–162. Institute of Electrical and Electronics Engineers Inc., dec 2015.

[54] Github - node-red/node-red: Low-code programming for event-driven applications. https://github.com/node-red/node-red. (Visitado em 16/12/2019).

[55] Particle. Particle Tutorials — Quickstart Guide. https://docs.particle.io/tutorials/iot-rules-engine/quickstart/. (Visitado em 18/12/2019).

[56] ThingsBoard - Open-source IoT Platform. https://thingsboard.io/. (Visitado em 18/12/2019).

[57] Seoyeon Kim, Jisu Park, Jaehyeok Jeong, Young-Sun Yun, Seongbae Eun, and Jinman Jung. Survey of IoT platforms supporting artificial intelligence. pages 65–66. Association for Computing Machinery (ACM), 2019.

[58] Riccardo Giorato. Real-Time Error Reporting — Catch Every Bug or Error, 2019. https://medium.com/better-programming/real-time-error-reporting-catch-every-bug-or-error-4c3df3b9a49d. (Visitado em 29/05/2020).

[59] Application Monitoring and Error Tracking Software — Sentry. https://sentry.io/welcome/. (Visitado em 27/05/2020).

[60] Platforms - Docs. https://docs.sentry.io/platforms/. (Visitado em 29/05/2020).

[61] Error Monitoring & Reporting Tool for App Stability — Bugsnag. https://www.bugsnag.com. (Visitado em 27/05/2020).

[62] The all-new Bugsnag is here — Bugsnag Blog. https://www.bugsnag.com/blog/all-new-bugsnag-launch. (Visitado em 03/06/2020).

[63] Rollbar - Error Tracking Software for JavaScript, PHP, Ruby, Python and more. https://rollbar.com/. (Visitado em 27/05/2020).

[64] @sentry/browser vs bugsnag vs rollbar — npm trends. https://www.npmtrends.com/@sentry/browser-vs-bugsnag-vs-rollbar. (Visitado em 27/05/2020).

[65] Duck DNS. https://www.duckdns.org/. (Visitado em 22/05/2020).

[66] home-assistant/home-assistant-js-websocket: JavaScript websocket client for Home Assistant. https://github.com/home-assistant/home-assistant-js-websocket. (Visitado em 22/05/2020).

[67] REST API — Home Assistant Developer Documentation. https://developers.home-assistant.io/docs/api/rest/. (Visitado em 22/05/2020).

[68] Node.js - Total.js Platform. www.totaljs.com. (Visitado em 07/02/2020).

[69] Rete.js. https://rete.js.org/. (Visitado em 07/02/2020).

[70] retejs/vue-render-plugin. https://github.com/retejs/vue-render-plugin. (Visitado em 07/02/2020).

[71] HTML Web Storage API. https://www.w3schools.com/html/html5{_}webstorage.asp. (Visitado em 25/05/2020).